The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

JULY/AUGUST 1987
VOLUME 5.4
$3.95

Starting Your Own Software House: Technical Support

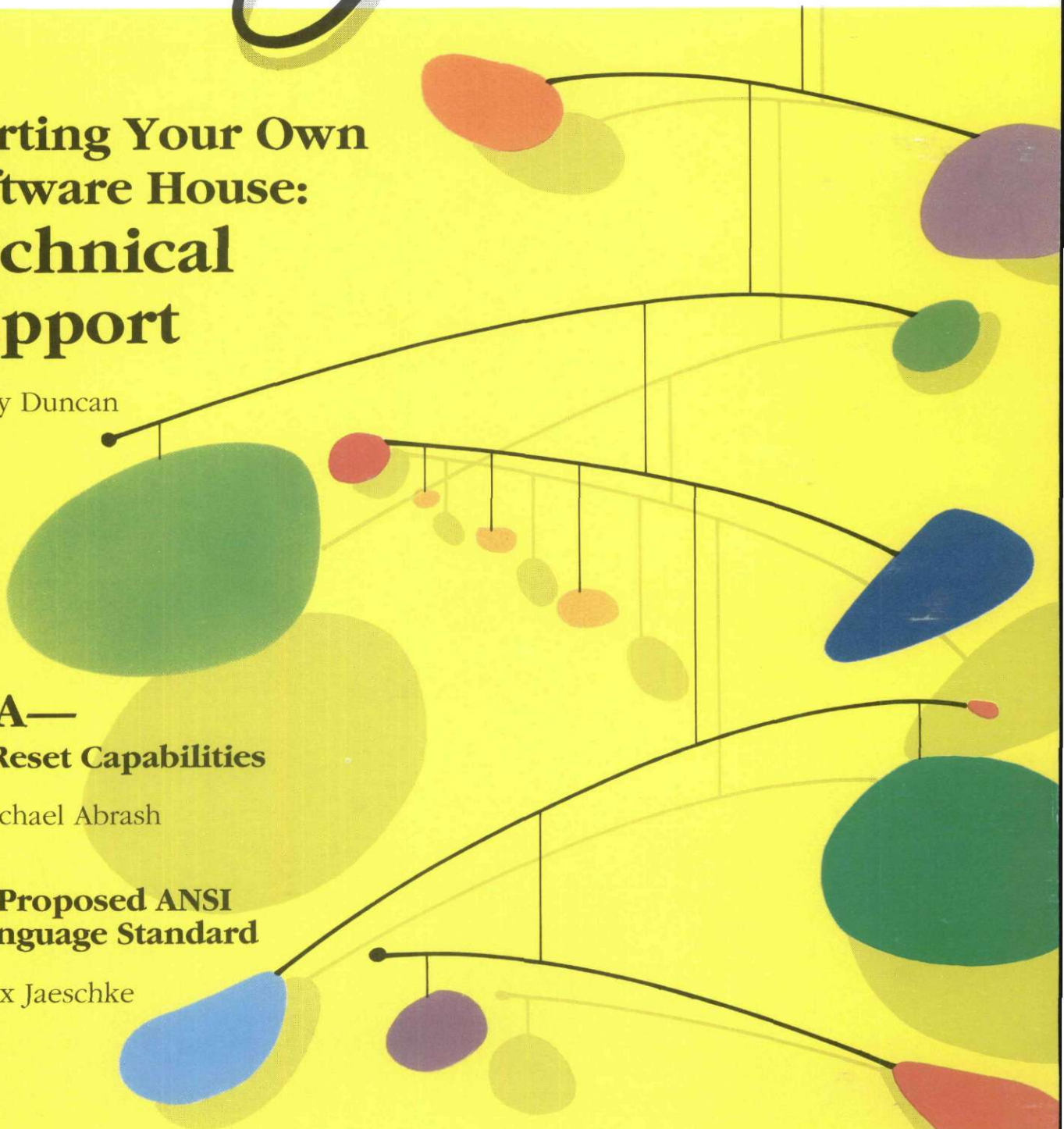
by Ray Duncan

EGA— Set/Reset Capabilities

by Michael Abrash

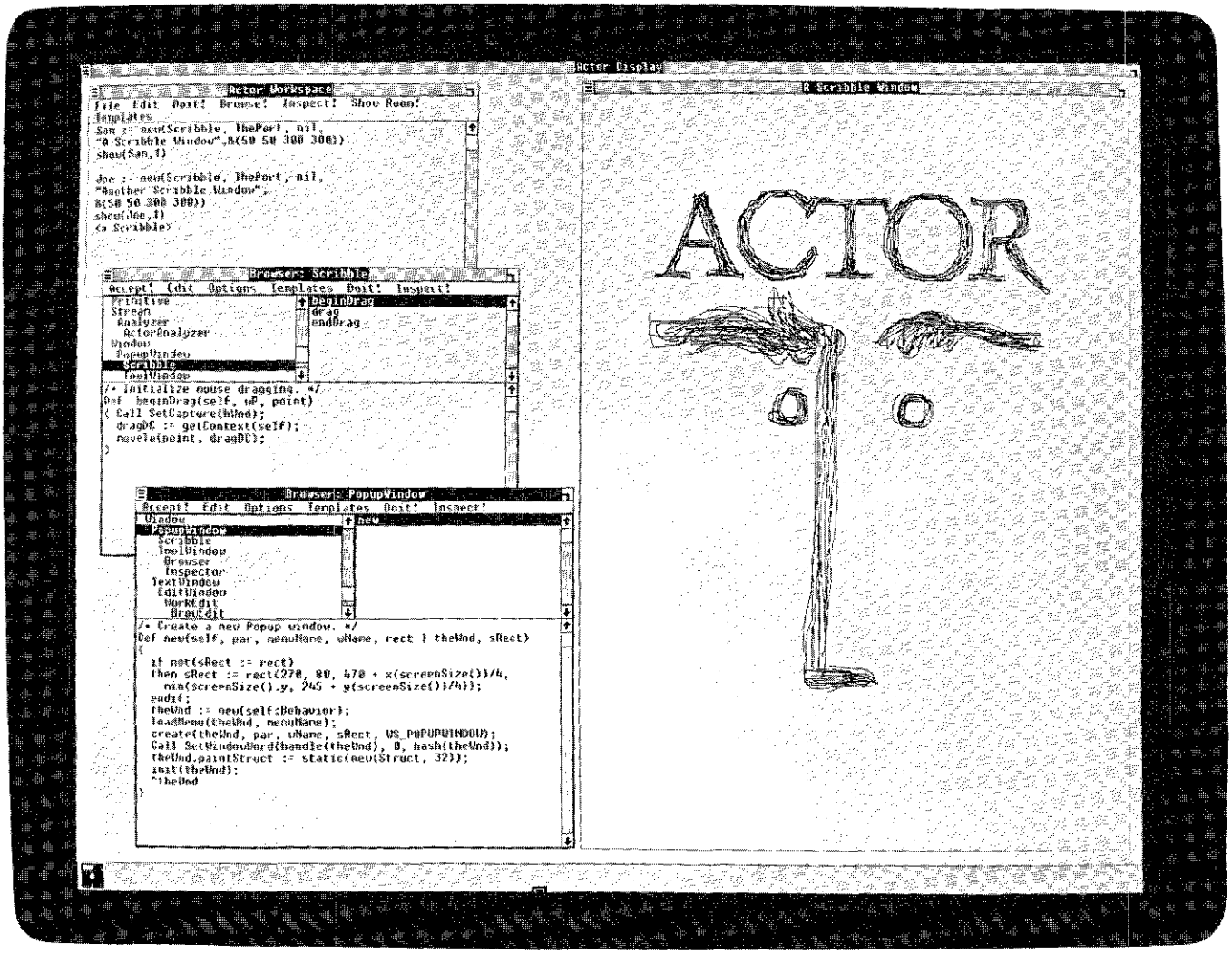
The Proposed ANSI C Language Standard

by Rex Jaeschke



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor. Call The Whitewater Group®, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT® WINDOWS

Number 73 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVERYTHING THE C PROGRAMMER NEEDS

Eco-C88 C Compiler:

A full, professional C compiler with many ANSI enhancements at an unbelievably low price.

* Prototyping, enum, void data types, plus structure passing & assignment * All operators and data types * Over 200 library functions * cc and mini-make for easy use * 8087 support * ASM or OBJ output * Lint-like tiered error messages * Fast code * CED editor (edit-compile-link from within the editor) * Expanded user's manual * Not copy protected * All for only $59.95!

Eco-C88 Flexi-Graph Graphics Package

Everything you need to write dramatic graphics effects into your Eco-C88 C programs.

* EGA, CGA, and Z100 support * Over 100 graphics functions (many are PLOT-10 compatible) * Most assembler support routines are outside small model code-data * Write thru BIOS (for compatibility) or to memory (for speed) * Graphics function help from CED editor * World, pixel or turtle color graphics modes * 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash, and fonts * Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual * Only $39.95

Eco-C88 Windowing Library

Use this library to build pop-up windows, help windows, selection menus, special effects—anywhere you need an attention getter.

* CGA and EGA support * Control any program that goes through the BIOS * Use up to 255 windows * No special window commands — use plain old printf( ) to write to a window * Resize and move windows * Custom window titles and borders * Can be used with ANSI device driver * Most window code-data are outside small model * User's manual and examples for only $29.95

Ecosoft Librarian

Combine your modules, functions, and subroutines into your own library for easy link commands. Compatible with any standard MSDOS OBJ files * Add, delete, and extract from a library * Get table of contents or index of a library * Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files * Complete with user's manual for only $29.95

Developer's Library

Contains the source code for for all library functions, including the transcedentals, memfiles, and those written in assembler. $25.00 with order, $50.00 if ordered later. (Sold only to Eco-C88 owners.)

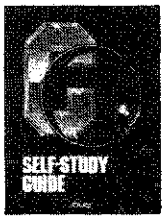
Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later.

C Programming Guide

(Purdum, Que Corp.)

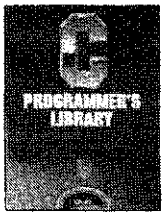
The second edition of the B. Dalton bestseller. Perfect for those just getting started with C. Includes discussion of many X3J11 ANSI Standards Committee recommendations. Many error messages from Eco-C88 give page references to this book. Price is $20.00 plus $2.00 shipping.



C Self-Study Guide

(Purdum, Que Corp.)

Using a question-answer approach, this book is filled with shortcuts, tips, techniques, and traps to avoid when learning C. Price is $17.00 plus $2.00 for shipping.



C Programmer's Library

(Purdum, Leslie, Stegemoller, Que Corp.)

Another B. Dalton bestseller. An intermediate C text for the programmer that wants to get the most from the language. Contains source code for many functions including an ISAM file handler. Price is $22.00 plus $2.00 for shipping.

To order, call or write:

1-800-952-0472 (for orders) or **1-317-255-6476** (tech. info.)

ORDER FORM CLIP & MAIL TO:
Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

- ☐ Eco-C88 C Compiler $59.95 _____
- ☐ Eco-C88 Flexi-Graph Graphics Package $39.95 _____
- ☐ Eco-C88 Windowing Library $29.95 _____
- ☐ Ecosoft Librarian $29.95 _____
- ☐ Developer's Library $25.00 ($50.00 if not with order) _____
- ☐ C Programming Guide $20.00 _____
- ☐ C Self-Study Guide $17.00 _____
- ☐ C Programmer's Library $22.00 _____

SHIPPING _____ $4.00

TOTAL (IND. RES. ADD 5% TAX) _____

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK

CARD# _____ EXPIR. DATE _____

NAME _____

ADDRESS _____

CITY _____ STATE _____

ZIP _____ PHONE _____

Number 8 on the Reader Service Card

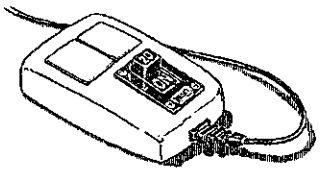
ECOSOFT

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

Volume 5.3
May/June 1987



Contents

4 **Note from the Editor**

6 **Letters**

12 **PJ Forum:** Starting Your Own Software House: Technical Support *by Ray Duncan*
The ins and outs of providing low-cost, time-saving support

18 **Redirections:** Unix Books, Part 3 *by Frank D. Greco*
The conclusion of Frank Greco's three-part survey of the literature on Unix

22 **Consulting:** Consultant Productivity and Effectiveness *by Paul Barkley*
Trim the unbillable hours and work more effectively

30 **On Graphics:** EGA Set/Reset Capabilities *by Michael Abrash*
The final aspect of data flow through the Graphics Controller

38 **C:** The Proposed ANSI C Language Standard *by Rex Jaeschke*
The type modifiers ***const*** and ***volatile***

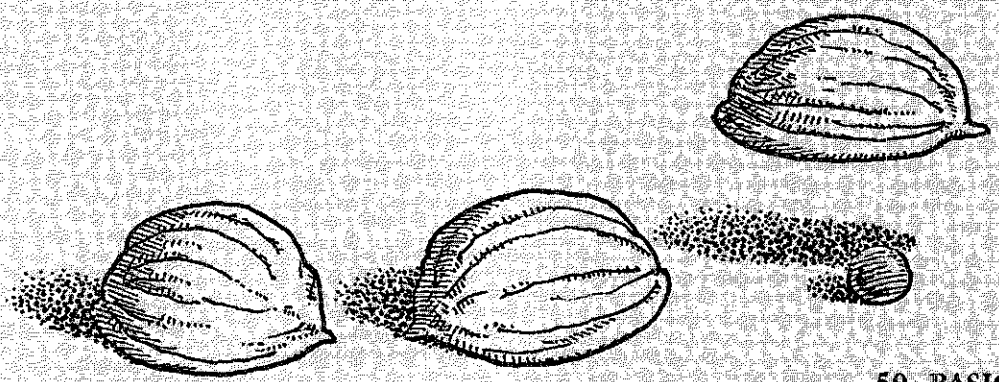
42 **C:** Character Tables and String Functions *by William J. Hunt*
Scan a number of strings quickly

44 **Utilities:** MWWIND—A Menu with Windows Utility *by Daniel A. Bronstein*
A utility to help a client use a hard disk

50 **BASIC:** Variable Passing and Hiding in QuickBASIC 2 Subprograms *by Bernard H. Robinson, Jr.*
Ways to deal with the global range of BASIC's variables

60 **Misc**

64 **Billboard**



50 BASIC

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

More Magic from Blaise.

Turbo C TOOLS™

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a library of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's attention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes Turbo C TOOLS the choice for experienced software developers as well as newcomers to C.

Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment; including removable windows, "side-kickable" applications, and general interrupt service routines written in C.

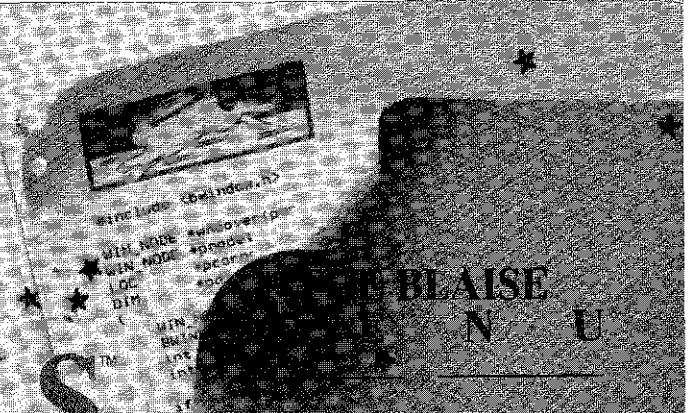
The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

- **WINDOWS** that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.
- **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.
- **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.
- **RESIDENT SOFTWARE SUPPORT** lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.
- **FAST DIRECT VIDEO ACCESS** for efficiency, and support for all monitors including EGA 43-line mode.
- **DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

In addition to Turbo C TOOLS, Blaise Computing Inc. has a full line of support products for Microsoft, Lattice and Datalight C, Microsoft Pascal and Turbo Pascal. Call today for details, and make magic!

Turbo C TOOLS supports the Borland Turbo C compiler, requires DOS 2.00 or later and is just $129.00



BLAISE N U

Turbo POWER TOOLS PLUS *$99.95*
Screen and cursor management including EGA 43-line support, DOS memory control, ISRs, shell interface, overlay code and much more. For Turbo Pascal.

Turbo POWER SCREEN
COMING SOON! General screen management, paint screens, block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM1 ports, I/O buffers up to 64K, XON-XOFF, up to 9600 baud, modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling, graphics routines, memory management, terminal program control, DOS file support and more. For MS Pascal.

C TOOLS PLUS *$175.00*
Windows, ISRs, screen handling, multiple monitors, EGA 43-line text mode, direct screen access, DOS file handling and more. For MS and Lattice C version 3.00 and later.

LIGHT TOOLS *$99.95*
Windows, ISRs, EGA 43-line text mode, direct screen access, DOS file handling and more for the Datalight C compiler.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports, I/O buffers up to 64K, XON-XOFF, up to 9600 baud modem control and XMODEM. For C or MS Pascal.

VIEW MANAGER *$295.00*
General screen control, paint screens, block mode data entry, field-by-field control with instant screen access. For C or MS Pascal.

RUNOFF *$49.95*
Text formatting for all those memos, books, printer control, page definition and more. Implemented in general purpose routines for inclusion in your own programs. Written in Turbo Pascal.

EXEC *$95.00*
NEW VERSION! Program chaining, memory save, DOS environment control, command line parameter passing and more. For Turbo Pascal.

TO ORDER CALL TOLL FREE
800-333-8087
TELEX NUMBER 338149

YES! I want to make magic!
Enclosed is $_____ for _____ copies of _____
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (_____) _____
Address: _____
City: _____ State: _____ Zip: _____
VISA or MC#: _____ Exp. Date: _____

Turbo C is a trademark of Borland International.

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316, Berkeley, CA 94710 (415) 540-5441
Number 2 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

Editors
Charles Aylworth, Erin O'Connor
Technical Editor
M. Steven Baker
Contributing Editors
Frank D. Greco, Tom Swan

Contributing Writers
Michael Abrash
Paul Barkley
Daniel A. Bronstein
Ray Duncan
Frank D. Greco
William J. Hunt
Rex Jaeschke
Bernard H. Robinson, Jr.

Publisher
Elizabeth Oakley
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Subscriptions
Gary Hoyer
Circulation
Preston Moser
Founder
Greg Estes

Business and Editorial Offices:
Programmer's Journal
29 West 29th Ave., Suite 5
Eugene, OR 97405
(503) 484-2162

Change of Address:
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403
(503) 484-2162

Subscriptions:
1 year (6 issues) $24 U.S. ($30 in Canada and $40 elsewhere)
2 years (12 issues) $44 U.S. ($55 in Canada and $75 elsewhere)
3 years (18 issues) $64 U.S. ($80 in Canada and $110 elsewhere)

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.
Advertising: For ad rates, deadlines, and placement, call Robert Tappan, Ralph Bentley, or Gary Hoyer (503) 484-2162, or write to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.
IBM and Personal Computer are trademarks of International Business Machines, Inc.; dBASE II and dBASE III are trademarks of Ashton-Tate.
Programmer's Journal (ISSN 0747-5888) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Eugene, OR, and additional mailing offices.
POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

©1987 OAKLEY PUBLISHING COMPANY

NOTE FROM THE (TECHNICAL) EDITOR

PJ Editor Departs

I'm standing in for Editor Robert Keller, who will depart from *PJ* after this issue. Robert has been with *Programmer's Journal* for almost two years, during which he nurtured the magazine into a solid technical journal. He stalked those wild computer jungles and bulletin boards for writers and article material. He waded hip deep through piles of news releases, new product announcements, and publications for those "gems" that might be of interest to you. He connived to get good writers and used a combination of threats and cajolery to get them to send their manuscripts in before deadline. He did an excellent job. I speak for the many *PJ* writers who are very grateful to have been able to work with him.

CIA Finds Bug in 80386

Intel (not really the CIA) found the inevitable bug in the current masks of the 80386. The problem occurs in some instances when performing multiplication and division in the 80386 mode on 32-bit registers. Intel has revised the chip mask; however, there will be delays before the new revision is available. This may delay new 386 machines unless the older revision chip is used. Good thing the current crop of machines are only running DOS programs and not maintaining your bank statements.

Semiconductors in the Courts

Some big semiconductor manufacturers are in the courts these days fighting over manufacturing rights to the 32-bit Intel 386 and Motorola 68020 CPU families. Intel and Advanced Micro Devices (AMD) had an agreement to share technology and for second sourcing for the 8086/80286 family. This agreement extended also to the 80386 chips; however, Intel would prefer to retain complete control of manufacturing and has refused to turn over the masks for the 80386/80387 and 80287 chips to AMD for over a year. They say that AMD has not designed enough peripheral chips for the 86 family to satisfy the pact. AMD claims it is losing $1 million a week in 80386 revenues. Meanwhile, Motorola is dueling it out in the French courts with Thomson-CSF, with whom it has a similar pact, over the 68020 chip family. Thomson has a hearing over its $525 million breach of contract suit scheduled for June.

IBM PC/2, OS/2 and Beyond

The microcomputer business seems to be in a funny mental state. After IBM's big announcement on April 2, there are many puzzled looks on people's faces. IBM did announce and demonstrate some "new" hardware and promised more to come. Personally, I was not terribly impressed with the new hardware, hoping for something more revolutionary. The new software, OS/2, remains "vaporware" yet to come. Without it, the vaporware hardware isn't of much use. The "new" IBM graphics standard wasn't even state-of-the-art last year. The new IBM machines come with 3" floppy disks and can be configured with 5-1/4" diskettes only by using external drives and power supplies (sorta like the old APPLE II with a drive hanging off the side). The new IBM box is not tall enough to take a 5" drive. Meanwhile, Compaq is having a hard time meeting the demand for their Compaq 386 machine. I use one at work these days maintaining large databases.

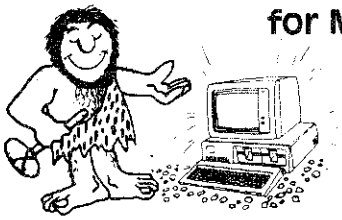
On the other side, many software developers are puzzled. Is it worth starting a new project now for MS DOS or for OS/2? When will OS/2 really arrive? Is the MS DOS operating system at the end of its life cycle? As the world turns. . .

—Steven □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Professional Programming Products

for Microsoft C, PASCAL, FORTRAN, and Assembly Language



FREE!
PC-WRITE™ text editor

PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!
NO ROYALTIES REQUIRED

ASMLIB
The Programmer's Library

- A linkable library with over 210 assembly language subroutines which are directly CALLable from Microsoft™ Assembler, C, PASCAL, and FORTRAN.
- Virtual disk file handling.
- Text WINDOWing allows up to 64 overlapping windows.
- Int. driven asynch. support.
- Graphics on EGA, herc. and CGA.
- Floating point math and trig outlines with 8087 support.
- Installable keyboard activated programs are easily written with ASMLIB's special functions.
- Plus much, much more.
- Supplied with complete source code.

Only $149.00 Complete

asmTREE
The Programmer's B+Tree Data File Management System

- A complete single/multiuser database development system written in assembly language and CALLable directly from Microsoft™ Assembler, C, FORTRAN, PASCAL, and LATTICE™ C programs.
- Up to 256 users.
- Up to 256 index and data files.
- Multiple key types.
- Multiple indices per index file.
- Duplicate and variable length keys.
- Read DBASE data files directly.
- Plus much, much more
- Supplied with complete source code.

Only $395.00 Complete

NEW

FPLIB
FPLIB™-IEEE Floating Point for REALIA™ COBOL!

- A full IEEE floating point math package for your REALIA COBOL programs.
- 8087 numeric co-processor support and EMULATION. Switching from emulation to true 8087 instructions is done automatically at RUN time - transparent to the user!
- A full set of both arithmetic and TRIGONOMETRY functions have been provided.
- Conversion routines allow your REALIA program to pass data to and from FPLIB by either using DISPLAY types or COMP-5 data types.
- FREE Updates for 1 year.
- SOURCE CODE available.

Only $149.00 Complete

NEW

Turbo.ASM
For TURBO Users!
Turbo.ASM™ (turbo dot a.s.m.)

- A unique programming tool which is a must for every Turbo Pascal user. The only package designed for the Turbo user who wants to get into Assembly Language.
- Outlines 4 different ways to add assembly language routines to Turbo programs without effecting your code space.
- Fully explains internal workings of Turbo Pascal and data passing methods.
- Includes a library of Assembly routines which can be used directly from Turbo Pascal.
- NO ROYALTIES required.

Only $99.95 Complete

B C ASSOCIATES
3261 No. Harbor Blvd., Suite B
Fullerton, CA 92635
1-800-262-8010
in Calif. Call
(714) 526-5151

Call for information on complete line of SimpleNET networking products.

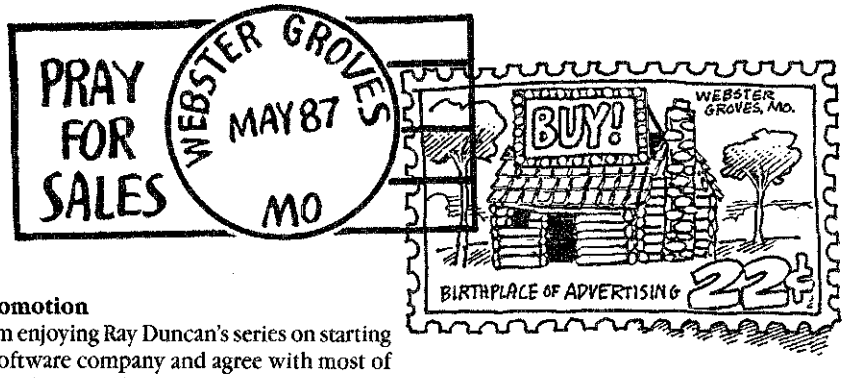
Outside CA, call TOLL FREE 1-800-262-8010

USE YOUR VISA OR MASTERCHARGE -

All prices include UPS shipping within continental United States. Outside U.S. please add $10 per package. Calif. residents please add 6.5% sales tax.

Number 36 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Promotion

I am enjoying Ray Duncan's series on starting a software company and agree with most of what he says about promoting a software product.

My company specializes in producing marketing communications for technical firms, so I can offer the following additional advice:

Public Relations: Write your press releases and product announcements exactly the way you want them to appear. Include a quotation from the author or a user that will spark interest. If it is interesting and well written, your press release will receive minor, if any, editing.

Submit articles to trade publications. Write articles that deal with some problem or difficulty that your product solves. Include examples or case studies.

Advertising: The ad must look professional because quality advertising implies a quality product. Always show the product, or examples of what the product can do. Emphasize benefits, not features. (Mr. Duncan said this, but it deserves to be repeated because very few advertisers follow this rule.)

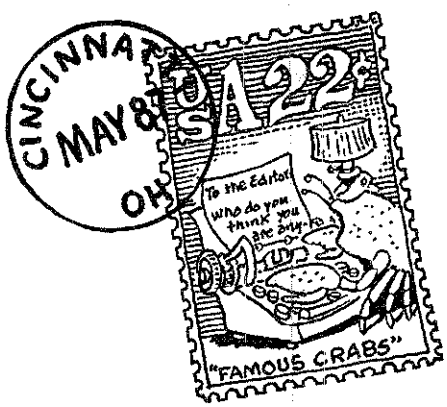
Use pictures or illustrations, and write the captions carefully. Remember, most readers read the captions first, the headline second, and then if they're still interested, the body copy.

Tell the customer what you want him or her to do—"Write for more information," "Call today to order," "Send for free demo," etc. Don't rely on the customer to figure out what you want him or her to do.

Brochures: The sales brochure should be considered an on-premise selling tool. Like the advertising, it has to look professional, imply quality, and give the customer a clear idea of the product's potential benefits. This is the place for your body copy—explain your advertising claims and convince the customer that he or she needs your product. Make it easy for the customer to buy—include an order form.

Smart Graphics is a marketing communications firm that specializes in advertising and design for technical firms. One of our software clients is the graphic design program Showoff for the Zenith Z100. You can see the ads we have designed and produced for Showoff in current issues of *REMark* and *Sextant* magazines.

Janet M. Hirsch
Smart Graphics
Webster Groves, MO



What's Appropriate?

If I buy your back page, can I get your cover, too? As well as several pages of editorial? Let other magazines give product reviews. If you want to test something, fine, but a product demo is not adequate. First Kahn did Lightning with *PC*, and now Turbo C with you. *Please* spend the time and effort on your generally excellent articles.

David and others
Cincinnati, OH

We welcome opportunities to review new products, the earlier the better.—Ed.

Advertising

As someone who has spent the last 20 years in marketing and advertising, I have been reading Ray Duncan's articles on "Starting Your Own Software House" with great interest. Almost everything he writes rings true with my experience in direct marketing. However, one statement could be misleading. The statement I question is, "Computer

HELP
is at hand

HELP/Control™ — an on-line help system for the IBM-PC. **HELP/Control** includes **HELP/Runtime, HELP/Popup** and our help screen compiler.

With **HELP/Runtime**, a few simple subroutine calls add context sensitive on-line help to your application. **HELP/Runtime** includes tested interfaces for C (Microsoft and Lattice), Pascal (Microsoft and Turbo), IBM BASIC (Interpreter and Compiler), Microsoft FORTRAN, COBOL (IBM and Realia) and assembler.

Use our concise screen definition language to build your help files. You define the bold captions on your help screens and specify the links to other screens. If you have existing documentation files, we supply a program which automatically marks them up to get on-line quickly. You can put an entire user or reference manual on-line, completely accessible to the user at all times.

HELP/Control also includes **HELP/Popup**, which provides memory resident access to your custom help screens. Use it to document dBase and 123 applications. **HELP/Popup** uses the same help files as **HELP/Runtime** and operates the same from the user's point of view, allowing you to provide a consistent on-line environment across diverse applications.

The complete package (software, on-line manual, printed manual, and demo programs) costs $125.00 and includes a royalty-free license to add **HELP/Runtime** to your applications and a license to make 25 copies of **HELP/Popup.** A demonstration diskette, including the on-line manual, costs $15.00. To order, or for more information (including dealer, multiple-copy and site-license pricing) call MDS at 207/772-5436. We accept MasterCard and VISA.

New options with release 1.20:

- **Use Dan Bricklin's Demo Program to build your help system. Our translator turns it into a HELP/Control source file.**
- **Include graphics in your help screens.**
- **Mouse support.**

MDS

MDS, INC., P.O. BOX 1237, PORTLAND, MAINE 04104

Number 69 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

magazine subscribers seem to have a remarkable mental filter for optical input that has not yet been discovered by neurologists; they simply don't register an advertisement until it has appeared in the same place for several months running."

This is not the case. The problem is *not* with computer magazine subscribers, but with the ad itself. Each direct response ad must stand on its own. Magazine ad salesmen who claim an ad has to run 3 to 6 months in a row to give it a fair chance to work either don't understand direct response advertising or are dishonest.

For example, what do you think the response would be if you ran a ¼ page ad with this headline:

> Introducing a Major New Program: I II III. The first 100% Compatible 1 2 3 Clone—FREE to the first 5,000 PC Users Who Call 1-800-555-1212!

I guarantee you would be overwhelmed with the response from this ad the very first time it ran. *Computer magazines can deliver tremendous response if you can deliver a powerful ad.*

Here are some guidelines for creating effective direct response advertising:

1. Create the ad in such a way that the very best prospects must see it! Make the ad look like an ad for what you are selling. Be prospect-specific. If you run an ad that gets the attention of 80% of all the subscribers to a magazine but only 5% of the best prospects for your product, your ad is not as good as one that attracts only 5% of the subscribers but 80% of the best prospects. Go after the attention of the prospects.

2. Make your ad large enough and strong enough to do the job you intend to achieve. You can have different goals for an ad. You might want to sell the product by mail; you may want to generate requests for more information; you may even just want to generate retail sales for the product. Whatever your goal, the ad must be big enough and strong enough to do the job—or it is a waste of money. The ad has to be big enough to attract the attention of your best prospects and strong enough to tell a selling story. You make ads stronger by offering introductory savings, a money back guarantee, no copy protection, free demo disks, testimonials, review quotes, and so on. You need a very strong ad to sell product directly by mail. However, a much smaller ad can be used to generate requests for more information. You have to decide which approach is the most cost effective.

3. Give your ad the best shot the first time. If you run a cheap little ad in a small publication and it doesn't work, you still don't know much. Perhaps the ad would work in a larger publication. If you have a new software tool for the C language, run your ad in the *C Journal*; make it large enough to accomplish the job intended, and run it in the strongest month if possible. Direct marketers know that some months are much better than other months. June, for example, may pull only 67% of the responses the same ad would get in January. (What magazine space salesman is going to tell you not to run an ad in the summer? More likely, such a salesman will tell you to run a series of 6 ads to give the product a chance.)

4. Run ads as long as they're paying their way. If an ad produces a profit for you, keep running it until it doesn't. In the meantime, test, test, test. Try to produce a better pulling ad; but keep the proved money-makers running. If two different ads in the same issue of a magazine both pay their way, run them both. If one ad pulls 100 responses and the other only 25 but you need only 20 sales to be profitable, still run both ads. Also, even if two magazines have a 95% overlap in circulation, use both magazines if they both produce a profit.

5. Test and keep good records. Making an ad 50% larger does not increase reponse by 50%. If a selling story can be told in 1/2 page, you may get almost as much response from the ½ page ad as you would from a full

TASKVIEW

ROCK SOLID MULTITASKING!

Packed with the power you need, TASKVIEW takes you beyond the limits of DOS.

- Communicate while you edit
- Compile while you print
- Load up to 5 megabytes of programs
- Manage resident utilities
- Time-slice multiple jobs
- Cut & paste between programs

TASKVIEW lets you load up to 10 of your favorite applications and switch between them at a keystroke. They can even continue to run while you work on something else! TASKVIEW is the BEST multitasker you can buy. Just ask our customers.

"Thanks for a wonderful product!"

"TASKVIEW is the only multitasker I own which runs efficiently & correctly both on my Zenith Z-151. . . and on my Orchid 286e . . . I also have DoubleDOS, DESQview, TopView, Windows, and Concurrent PC-DOS".

"Fantastic! What DoubleDOS, MS-Windows, and all the other 'stuff' should have been!"

"My BBS is now up 24 hours, 7 days a week thanks to TASKVIEW."

"Thanks! I needed that!"

TASKVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:

(800) 367-0651

or send $79.95 + $3.00 S&H ($8.00 Intl.) to:

Sunny Hill Software

PO Box 55278, Seattle, WA 98155-5278

For more information call (206) 367-0650

DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 85 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

page ad. An ad that fails in July could be profitable in February. A product that won't sell directly from the ad may sell very well from the information you send in response to a request. One headline may pull 19 times more response than another headline—even though the body copy is the same. You must test, keep good records, and experiment with different approaches. Use other people's experience. Look at the last 2 years of a magazine's issues. Look for ads for products similar to yours that run month after month. Run where and when similar products have found success. (Usually, direct marketers don't run ads that don't work for very long.)

6. Read the best material on direct marketing. The more you know about direct marketing, the better your chance of success. The best book on direct marketing is Bob Stone's *Successful Direct Marketing Methods*. Read this book and any others you have time for.

There are more than 500 ways to improve a direct response advertisement. These are only a few of the most important. Get the attention of your best prospects, deliver a message strong enough to elicit a response, make each ad prove itself, and be as cost effective as possible.

Vince Mooney
Marketing Director
Condominium Home Realtors
Tulsa, OK

Continued . . .

DeSmet C
SUPPORTS
ROMABLE CODE

DeSmet C *$109
- Data & Stack in RAM
- Code executes out of ROM
- Full K&R + V7 extensions
- Inline **asm**
- Assembler, linker, librarian
- Full screen editor (SEE[tm]),
- 8087 & S/W floating point

DeSmet C with D88 Debugger $159
- Set Breakpoints by line number or function name
- Examine Global/Local variables by name
- Show C source while debugging
- Both D88 and User displays

Large Memory DeSmet C w/ D88 $209
- 32-Bit Pointers (Full Megabyte Addressing)
- Fast Access of Static Data
- Includes the standard compiler features mentioned above.

*Both D88 & Large Memory available as options at $50 each.
We also have a C compiler for the Mac. $150. Call for details.

— *plus* —

Graphics $35
BASICA-like (+src)

Hacker $25
Source for start-up, RAM.COM & more

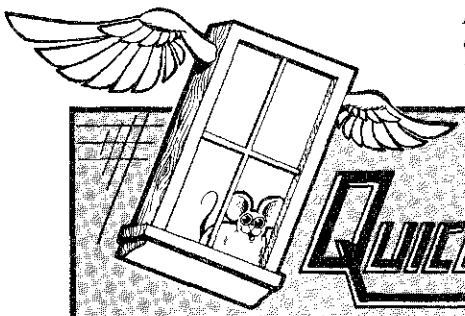
Make $50
Full UNIX-level

Tools (+source) $35

XARRAY $39
Large Arrays (+source)

C WARE
CORPORATION
P.O. Box C
SUNNYVALE, CA 94087 USA
(408) 720-9696 TELEX: 358185
Street Address: 505 W. Olive, #767
VISA, MC & AMEX accepted

Number **38** on the Reader Service Card



QuickWindows™

Give your BASIC programs a friendly user interface with *QuickWindows* ™

- ✓ Fast! Written entirely in assembly language.
- ✓ High level window management to save you lots of programming time. Popping a window is easy as (1) WOPEN, (2) WPRINT, (3) WCLOSE. Contents under window is automatically saved and restored.
- ✓ Customize your own titles and borders. Each window has its own cursor and attributes, just like the regular display.
- ✓ Full mouse interface for Microsoft compatible mouse.
- ✓ High level pop-up, pull-down, and help menuing also included. Can be used with keyboard and/or mouse.
- ✓ Over 60 functions help you get the most out of your windowing and menuing needs, including WOPEN, WCLOSE, WCOLOR, WPRINT, WLOCATE, WCLS, WMOUSE, POPMENU, HELPMENU, MENUBAR, MENUSET, MENUON, and more! Many examples included!
- ✓ No royalties required for using Quickwindows™ in your programs and not copy protected. 30-day money back guarantee. For IBM/True compatibles and Microsoft QuickBASIC or BASCOM, DOS 2.0 or higher.

Order NOW! Use your VISA/MC and call (401) 274-LINK

❑ Yes, please send me QuickWindows w/o source for $59
❑ I would like QuickWindows with assembly source for $79
Add $4 Postage and Handling. RI residents add 6% sales tax

Name _____
Address _____
_____
City _____
State _____ Zip _____

Send Check / MO to:
Software Interphase, Inc.
5 Bradley Street, Suite 7
Providence, RI 02908

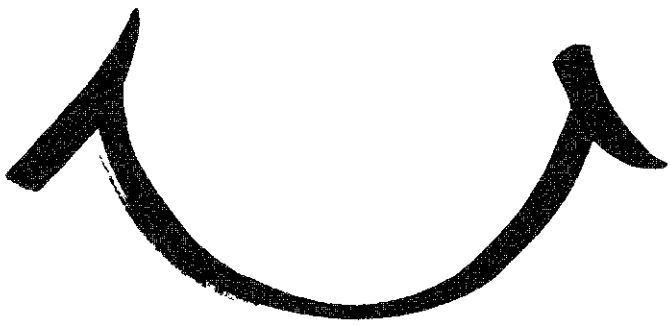
Dealer Discounts Available

MasterCard VISA

Number **115** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

COMPUTER FAIRES® GIVE YOU AN UNFAIR ADVANTAGE.



The Northeast and West Coast Computer Faires are going to help you surge ahead into new business opportunities in 1987 and 1988. Attendees will find the latest and most innovative products and services to put them ahead in the quest for productivity. Exhibitors searching for a sophisticated audience that's ready to buy will be smiling at their extra sales.

And extensive promotion, a full conference series, and special features like State-of-the-Art Professional Development Seminars help add up to give you the Computer Faire advantage—a smile of triumph.

Attention exhibitors! To reserve exhibit space or to receive more information call (617) 449-6600, Ext. 4013.

Northeast Computer Faire '87
October 15–17, 1987, World Trade Center, Boston

West Coast Computer Faire '88
May 5–8, 1988, Moscone Center, San Francisco

I want the Computer Faire advantage.

Send me ☐ exhibitor ☐ attendee information on ☐ Northeast ☐ West Coast Computer Faires

PJ 7/8 '87

Name_____

Title_____

Company_____

Address_____

City_____ State_____ Zip_____

Telephone ( )_____

Return to The Interface Group, Inc., 300 First Avenue, Needham, MA 02194

Number **58** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PORTABLE POWER WITH MasterFORTH

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest.

If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in assembler to make it even faster. MasterFORTH's relocatable utilities and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint and trace your way through most programming problems. A string package, file interface and full screen editor are all standard features. And the optional target compiler lets you optimize your application for virtually any programming environment.

The package exactly matches *Mastering Forth* (Brady, 1984) and meets all provisions of the Forth-83 Standard.

MasterFORTH standard package	$125
(Commodore 64 with graphics)	$100
Extensions	
Floating Point	$60
Graphics (selected systems)	$60
Module relocator (with utility sources)	$60
TAGS (Target Applic. Generation System) – MasterFORTH, target compiler and relocator	$495
Publications & Application Models	
Printed source listings (each)	$35
Forth-83 International Standard	$15
Model Library, Volumes 1-3 (each)	$40

(213) 821-4340

MICROMOTION

8726 S. Sepulveda Bl., #A171
Los Angeles, CA 90045

Number 13 on the Reader Service Card

More on TSRs

I was surprised in the discussion of TSR programs and in the letter from Andy Canfield (*PJ*, Vol. 5.2) about FUN 50h and 51h *not* to find the following key point even mentioned. Namely, if the TSR program does any allocating or deallocating of memory via the 4Ah, it *must* avail itself of the services of FUN 50h/51h to set the active PSP to its own PSP before the call. Otherwise, DOS allocates/deallocates memory to the program that has the current active PSP. In other words, the Microsoft documentation is incomplete at this point. Not only must the segment register ES point to the PSP of the TSR, but the active PSP must be reset as well.

I ran across this problem in trying to make a call to the EXEC function 4Bh from a TSR. I had the problem that when the application program terminated, DOS declared the memory belonging to the TSR as free. Since the TSR was invoked by a hot key, this led to a system crash on the next invocation.

Here is a chunk of code out of the EXEC code of the TSR to illustrate what I mean. The TSR was loaded with FUN 31H and claimed 70K (or more) of memory. It now wants to deallocate a portion of that to call up a second command processor. Here is the relevant code:

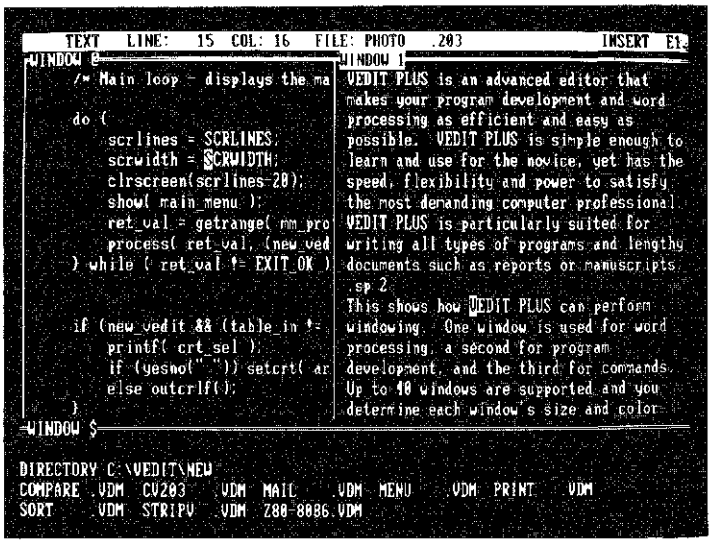
```
        ;DOS equates
        SET_CURRENT_PSP equ 50h
        GET_CURRENT_PSP equ 51h


        ;----------------------
        ;Get the address of PSP of the calling program and save
        ;it. Then reset the current PSP to our own.
        ;----------------------
        GET_USER_PSP_ADDRESS:
                mov ah,GET_CURRENT_PSP
                int 21h
                mov USER_PSP,bx

                mov bx,cs
                mov,ah,SET_CURRENT_PSP
                int 21h
        ;----------------------
        ;Find out where a command.com is supposed to be.
        ;----------------------
                call GET_COM_SPEC
        ;----------------------
        ;The minimum amount of space for us, so there is room for a
        ;second command processor to load.
        ;----------------------
        ;       etc.

        EXEC:   mov,ax,4b00h    ;Function code to EXECute a program.

                int 21h         ;At the return, all regs are
                                ;loused up,
OLD_DOS:        cli
RESTORES:       mov ss,cs:STACK_SEGMENT
                mov sp,cs:STACK_POINTER
                sti

                mov bx,cs       ;so restore them, with careful
                mov ds,bx       ;attention to the stack.
                mov es,bx

        ASSUME DS:CODESEG,ES:CODESEG
;----------------------
;Make sure ours is the active PSP
;----------------------
                mov ah,SET_CURRENT_PSP   ;bx is pointer for the
                int 21h                  ;SET_PSP call.

;----------------------
;Recover the memory we gave up for the shell.
;----------------------

                        mov bx,0ffffh   ;ask for all possible memory
                        mov ah,4ah      ;at the return BX contains what
                        int 21h         ;is available, so take that
                        mov ah,4ah
                        int 21h

                        mov ah,1ah      ;restore caller's DTA address
                        lds dx,CALLER_DTA
                        int 21h

                                etc.
```

Another important point illustrated in the code snippet above is one I cannot remember ever seeing discussed, least of all in the documentation from Microsoft. A call to the EXEC function will change the current DTA to 80h in the new PSP, and the EXIT routine does *not* RESTORE it. If a program relies on a DTA's being set once at the beginning of a session and not being reset before each use, a call to the EXEC function can have disastrous effects.

Larry Smith
Mathematics Department
University of Virginia
Charlottesville, VA

In general, TSRs allocate the memory they need when they're invoked. Although it's possible to allocate and deallocate memory later, you've pointed out some of the problems—why TSRs don't usually allocate/deallocate memory later. The same problem comes up when opening files. For this reason, DOS TSRs (PRINT.COM) traditionally use File Control Blocks (FCBs) rather than file handles.—Ed. □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


TEXT LINE: 15 COL: 16 FILE: PHOTO .203 INSERT E1
WINDOW 2
/* Main loop - displays the ma
do {
scrlines = SCRLINES;
scrwidth = SCRWIDTH;
clrscreen(scrlines-20);
show(main_menu);
ret_val = getrange(nm_pro
process(ret_val, (new_ved
} while (ret_val != EXIT_OK)
if (new_vedit && (table_in !=
printf(crt_sel);
if (yesno("")) setcrt(ar
else outcrlf();
}
WINDOW 1
VEDIT PLUS is an advanced editor that
makes your program development and word
processing as efficient and easy as
possible. VEDIT PLUS is simple enough to
learn and use for the novice, yet has the
speed, flexibility and power to satisfy
the most demanding computer professional.
VEDIT PLUS is particularly suited for
writing all types of programs and lengthy
documents such as reports or manuscripts.
.sp 2
This shows how VEDIT PLUS can perform
windowing. One window is used for word
processing, a second for program
development, and the third for commands.
Up to 10 windows are supported and you
determine each window's size and color.
WINDOW $
DIRECTORY C:\VEDIT\NEW
COMPARE .VDM CV203 .VDM MAIL .VDM MENU .VDM PRINT .VDM
SORT .VDM STRIPV .VDM Z80-8086.VDM

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

(Call for FREE DEMO disk)

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS defies comparison.

Try A Dazzling Demo Yourself.

The free demo disk is fully functional - you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial - you experiment in one window while another gives instructions.

The powerful 'macro' programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a 'macro' - it shows that no other editor's 'macro' language even comes close.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. (Yes! We support windows on most CRT terminals, including CRT's connected to an IBM PC.) Order direct or from your dealer. $185.

Compare features and speed	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	100 +
Multiple file editing	20 +	2	No	20 +
Windows	20 +	2	No	20 +
Macro execution window	No	No	No	Yes
Trace & Breakpoint macros	No	No	Yes	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Configurable keyboard Layout	Hard	No	Hard	Easy
'Cut and paste' buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
On-line calculator	No	No	No	Yes
Manual size / index	250/No	42/No	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min.	34 sec.	1:07 min.	6 sec.
Pattern matching search	20 sec.	Cannot	Cannot	2 sec.
Pattern matching replace	2:40 min.	Cannot	Cannot	11 sec.

VEDIT PLUS FEATURES

- Simultaneousy edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 'scratch-pad' buffers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size.
- Optimized for IBM PC/XT/AT. Color windows. 43 line EGA.

EASY TO USE

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I, PASCAL, etc.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert to/from WordStar and mainframe files.
- Print any portion of file; selectable printer margins.

MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts, keyboard input, 17 bit algebraic expressions, variables.
- Flexible windowing - forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Menu-driven tutorial

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Starting Your Own Software House: Technical Support

by
Ray Duncan

Greetings, prospective software mogul! This is the third in a series of articles focused on the dull but vital mechanical details of running a small software house. The first installment covered business licenses, shipping, and bank accounts, and in the second we reviewed some methods of publicizing a new product. In this month's episode, we will reflect on the ways and means of technical support. I hope that my friends who are also LMI customers out there in *PJ*-land will try to be understanding if I sometimes sound a little calloused or cynical in order to make a point.

The Scenario

For months, you've been spending every spare minute of your evenings, weekends, and vacations in an intense creative effort: coding and debugging the next Great American Spreadsheet (or programmer's editor, or C graphics library, or B-tree index manager, or whatever). You've written the manual and had it printed, you've placed your ads, you've gotten some phone orders, and now—Happy Days!—you've shipped the first copies of your product. You've become a software vendor.

In a few years you'll look back at the day you dispatched your first package and realize that it marked the beginning of the end.

If you're not careful, in a few years you'll look back at the day you dispatched your first package and realize that it marked the beginning of the end, the end, that is, of your days as a blissful, undistracted, self-directed programmer—the state you were probably seeking when you went into business for yourself in the first place (those of you who are going into business primarily for the dough, to travel, or to make friends and influence people may want to just control-C out of this article right now).

Here is the crux of the problem: The more copies you sell, the more requests for support you will get. Each request has two costs: (1) the time required to communicate with the customer and resolve the problem and (2) the indirect cost of the disruption to your train of thought and the time required to get your mind back on whatever you were doing. At some point, the number of interruptions for technical support you encounter during a day plus the other inevitable time overheads of running a small business—paying bills, ordering supplies, going to the bank and post office, etc.—will interact and cross some threshold, and you will find that your programming productivity has essentially dropped to zero.

The support required per copy is of course highest immediately after your product is released, for several reasons. The early versions of your program are bound to have bugs, and the documentation will have holes and errors that will be gradually resolved. There is no existing base of experienced users to provide informal advice and caveats to new purchasers. And your customers are well aware that you are a small vendor, easily accessible, anxious to please, and in no position to be "difficult."

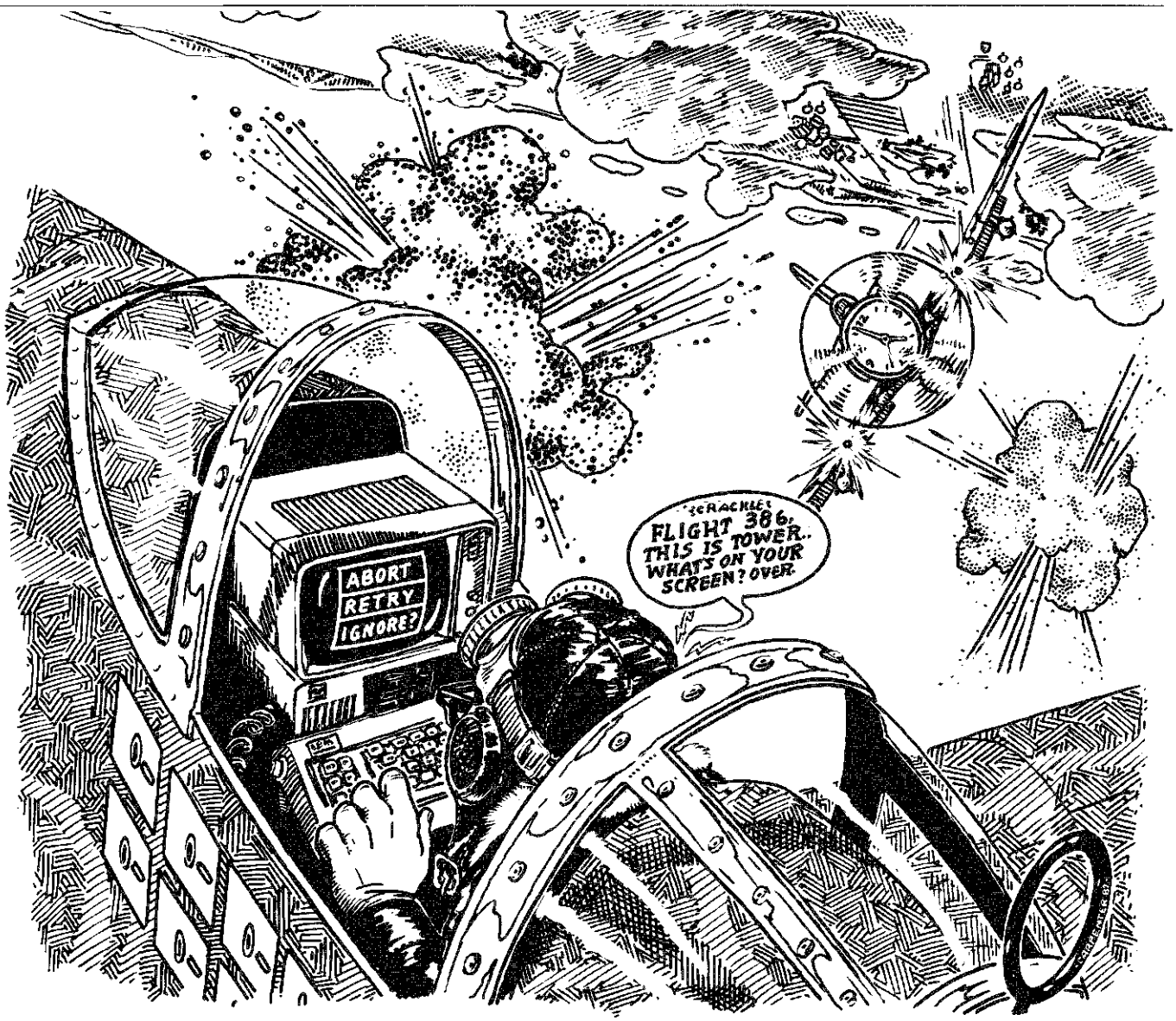
In the long run, if you listen to your users and can find time to constantly improve your product and documentation in response to their feedback, the support demanded per copy shipped should drop off steadily. There will be a point (perhaps one or two years after you go into business) at which the total technical support pressure will max out. If you survive this period with both your solvency and your sanity, perhaps making enough money to add an employee to handle the run-of-the-mill support questions and program maintenance, you'll once again be able to devote increasing proportions of your time to communion with your favorite computer.

Phone Support

Telephone technical support is a major factor in customer satisfaction and word of mouth advertising for your product. Yet every phone call infringes on the time you need to improve your product and stay abreast of the competition—or write new programs to broaden your product line and become less vulnerable. These contradictory statements have to be reconciled somehow, in a manner compatible with both your customers' happiness and your own peace of mind.

The most important consideration in telephone technical support, especially when you're working by yourself or with a very few employees, is close control over your time. When you answer the phone, you've got to encourage the customer to focus on his problem and avoid "chatting," help him to formulate the question clearly, decide whether it's something you can help him resolve, and bring the exchange to a rapid yet successful conclusion. The key is to keep the conversation moving and germane to its original goal; don't get diverted into philosophical dialogues, speculations on IBM's future products, or debates over your past implementation

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



decisions (unless you want to, of course!).

Since it's terribly difficult to concentrate on programming (or anything else, for that matter) if you are talking to customers about a totally unrelated topic for ten minutes out of every half hour, you may find it necessary to restrict their access to you in some way. I will describe two different methods that we have tried at my own company (with mixed success). Both methods are posited on the availability of a secretary or assistant to run interference on the phone; if you are running a strictly one-man show, these approaches aren't worth much. (Remember the classic joke: "I'm sorry, did I wake you up?" "No, I had to get up to answer the phone anyway.")

The first scheme, which we can term the "fixed hours method," is to establish a certain part of the day during which technical support is available and tell all callers outside that period to try again the next day. Choose the support hours according to your location so that all customers have a reasonable chance to reach you. For example, if you live on the West Coast, you can't decree technical support hours from 2:00 p.m. until 5:00 p.m. since that totally excludes business customers on the East Coast. Although the fixed hours method wipes out a certain portion of your day on a predictable basis, at least it leaves the burden of the phone cost on the customer.

The other gambit, which we can christen the "telephone call-back method," has some definite advantages. The basic strategy is to buffer everything: Have your secretary write down the name, phone number, program serial number and version, and a detailed problem description for each caller. The simpler questions can be dealt with by just writing the answer on the same message slip and having your secretary call the customer back—this protects you from getting drawn into long conversations ("By the way, as long as I've got you on the phone. . ."), and your secretary will quickly learn the solutions to the most common problems and field them automatically. Other, more complex, calls can be returned when it is most convenient or while you are doing something mindless (like printing a listing).

The big disadvantages of the call-back approach are the horrendous impact on your phone bill and the frequency with which you get involved in cycles of "phone tag." If the customer lives on the other side of the country, the chances are about 50/50 that when you try to call him back he either hasn't come back yet, is out to lunch, or has already left for the day ("you know those programmers").

A separate but important issue in telephone support is to establish the domain of problems for which you are willing to be responsible.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you are selling a programming tool, you will soon find that a high proportion of the calls have only a tangential relationship to your product. During one period, where we were providing support on a call-back basis and thus had fairly good records, I found that about only 10% of our calls were legitimate requests for support, that is, calls prompted by erroneous or missing documentation or by a genuine bug or other problem with the product. The remainder were about equally distributed between calls for information easily found in the manual, calls due to bugs in the customer's own code, or requests for help that ultimately boiled down to free systems-level consulting on 80x86 programming and/or MS DOS.

Newsletters

Newsletters are a great way for you to let your user base know about updates to new versions, bug fixes, price changes, and the like. They are also a nifty way to announce new products to a relatively receptive audience (assuming they are happy with your previous products)—particularly if you make your old customers feel appreciated by offering them special introductory discounts. To ensure that the newsletters won't be diverted directly to the round file, you need to supplement the duller bug and update information and the sales hype with a liberal dose of useful material, such as case histories of successful applications or handy tricks and work-arounds.

You can make up a very nice-looking newsletter with just about any decent word processor and a laser printer. Desktop publishing packages are all the rage, but they are complex, and unless you have an artistic bent and some experience with page composition and font selection, you can come up with some pretty gruesome results. Be sure to include a copyright notice and your full company name, address, and phone number on the first page and a running footer on each page that includes your company name again, the newsletter date, and the page number. Then if isolated pieces of your newsletter get photocopied and passed around or filed away, the eventual reader will know whom to credit (or blame).

A proportional, serif font such as Times Roman looks best for the body of the newsletter, and a sans-serif font such as Helvetica is more suitable for headings and titles. Use a non-proportional font, such as Courier or the Hewlett-Packard LaserJet "lineprinter" font, for all listings or examples of interaction with the computer—this makes it easier for the reader to interpret adjacent spaces and punctuation in commands or program statements. Otherwise, avoid the use of many different font styles and sizes—they are confusing to the eye and detract from the content.

The original of your newsletter will of course come off the laser printer single-sided. Have it reproduced double-sided by the quick print shop to save money and weight. If you are dealing in relatively small quantities (less than 500), it is usually cheaper to xerox; offset printing is more economical for quantities larger than 500. The print shop can also staple and fold the newsletters for you at a small additional cost (this will save you hours of tedious manual labor and is well worth the expense). It's then up to you to stuff the newsletters into envelopes, seal them, slap on the address labels, run each envelope through the postage meter or stick on a stamp, and drag them in bags to the nearest jumbo mailbox.

The optimum length for the newsletter will depend on how you plan to mail it. If you are going to mail first class, an eight- or ten-page newsletter (four or five sheets) is most practical. It turns out that four 8.5" by 11" sheets in a #10 business envelope fit nicely under the one-ounce limit for 22 cents postage. If you obtain a bulk rate permit ($50 per year), you can mail up to two ounces (about 9 sheets in a #10 envelope) for 12.5 cents, and you can have the permit number printed right on the envelopes to avoid stamping or metering. Note that bulk mail has to be sorted and bundled by ZIP code, can be accepted only at certain postal stations, and gets delivered when the Post Office gets good and ready.

If the vast majority of your customers are in the U.S., you can bypass the envelope stuffing step, and gain a little extra room for newsletter contents besides, by adding another sheet. Just leave the back page of the newsletter blank except for your return address, fold and staple the newsletter with the blank side out, and apply the address label and postage. However, newsletters destined for points outside the U.S. should always be put in an envelope and mailed first class. This increases the probability that they will arrive in readable condition (the probability is unfortunately still nowhere near unity). It costs 22 cents to mail one ounce to Canada and Mexico and 37 cents by surface mail to all other countries. You'll probably find overseas air mail rates prohibitive (88 cents per ounce) unless you charge your customers extra for this service.

The main benefit of using first class mail instead of bulk rate (aside from speed) is that the letter will automatically be forwarded if possible or returned to you if undeliverable, allowing you to keep your mailing list clean. If you use bulk rate, you will not get this service unless you have the legend "Address correction requested" printed on the outside of the newsletter or envelope, and then you will be charged 22 cents each for the corrections when they come back. This is still a good investment compared to the cost of sending additional newsletters into the void (see below).

Here are some ballpark expenses for newsletter production, based on our most recent mailing from LMI. Printing 10 pages (five double-sided sheets) costs about $400 per thousand sets. Stapling and folding are $30 and $90 per thousand, respectively. Envelopes printed with a return address and the bulk rate permit number cost about $30 per thousand. We hire teenagers at $5 per hour to stuff, address, and sort the newsletters and find that it requires about eight kid-hours per thousand newsletters out the door. The net cost per newsletter including postage comes out to around 72 cents per U.S. customer and about 96 cents per overseas customer.

One of the most important things you can do to make life easier over the long haul is to start building a customer database from the very first day you ship product. This database should include, at minimum, the user's name, company name, address, date of sale, and serial number and version. Other data fields will suggest themselves to you as time goes on, and if you use a robust database manager such as Microrim's RBASE, they will be easy to add at a later date without compromising the information that is already in the file. Every minute you spend updating the database will be repaid over and over as you use it to sort and print mailing labels for newsletters and announcements. Later, you may find it possible (as we did) to integrate the mailing list into a more comprehensive set of programs to track orders, print invoices, and validate calls for technical support.

Bulletin Boards

A dial-up Bulletin Board System (BBS) is an excellent way to provide technical support. It is a superior means of communication with your customers because it forces them to state their questions clearly (or at least concisely) and they can reach it at any time of the day or night. It also allows you to budget your time much more efficiently: You can sit down at the bulletin board once a day and solve everybody's problems in mental batch mode.

You can even improve the chances of your users' calling the BBS, instead of you, by making available a low cost "electronic technical support package." You can buy

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

1200 baud internal modems in quantities of ten for around $80 apiece, bundle them together with a good public domain or "shareware" communications program, and pass them on to your users at a minimal markup.

There are additional benefits to the use of a BBS that become apparent only over time. If you are traveling or on vacation, you can still respond to customer queries by dialing up your "home" system once a night with the aid of your portable computer and modem. When customers have problems with a program that can't be resolved by an exchange of messages, they can upload the offending code or data to you to look at and debug on your own machine. You can put program updates and patches on the BBS for customers to download, along with new product announcements and tutorial text files. And as your customers get accustomed to use of the BBS, they will start to help each other out with problems (reducing the load on you) and may also use it to exchange source code and handy utilities.

Although it is possible to run some BBS programs on a minimal floppy disk based system, they will all perform best with at least 512KB of RAM and a 20MB fixed disk. Just about any old PC clone, 80-column video adapter and monitor, and 300/1200 baud internal or external Hayes-compatible modem will do; an 80286-based machine is definitely not needed. You should be able to put together this equipment configuration for about $1000 by shopping the mail order houses or your local computer swap meets and flea markets. Getting a modem that will also support 2400 baud will add another $100 or so to the total cost, but will make your long-distance customers happier. Installation of an additional phone line for use by the BBS will cost about $120 up front and another $15-20 per month (business rates in my area—your mileage may vary).

Some small companies economize by using the same PC for invoicing or copying disks during the day and making the BBS available only outside normal business hours. I recommend that you avoid this computer-sharing if possible. Customers on the opposite coast or overseas will want to call the BBS during their evening hours, which are billed at lower rates, but this period is likely to collide with the working day in your time zone. Also, if the BBS is up around the clock on its own phone line, it will cut down on voice calls for technical support during the day.

There are a great many BBS software packages available, both public domain and commercial. Nearly all are menu-driven and support the same basic functions: reading and writing messages (public or private), on-line help, and uploading and downloading files. Some BBS programs offer additional features, such as support for conferencing, multiple phone lines, a programmable script or questionnaire facility that you can use to validate users or allow them to place orders on-line, and password or security level protection for file downloading.

The two most popular public-domain BBS packages are RBBS-PC and FIDO. RBBS-PC originated with the Capital PC User's Group and is floating around in scores of minor versions and variations; it can be obtained from the Capital PC Software Exchange, P.O. Box 6128, Silver Spring, MD 20906. FIDO is similar to RBBS-PC (though somewhat slower), but has one very interesting unique feature: All the FIDO boards in the world are registered at a central location and automatically pass messages to one another during the wee hours of the night, providing low-cost, world-wide electronic mail to their users! You can get more information about FIDO from Tom Jennings, Suite 118, 2269 Market St., San Francisco, CA 94114.

I ran a derivative of RBBS-PC on my own company's technical support bulletin board for over a year, but eventually decided that I would be better off paying for the quality, support, and documentation that come with a commercial package. After dialing up and trying out boards based on many different BBS packages and quizzing many BBS operators ("Sysops"), I finally settled on PC-BOARD, a product of Clark Development Company, P.O. Box 71365, Murray, UT 84107, (801) 261-8974.

PC-BOARD is available in a public domain version (Version 10) and four commercial levels (Version 11): PC-BOARD 11/S (Standard, no remote DOS functions or networking, $79.95), PC-BOARD 11/D (Deluxe, remote DOS access but no networking, $119.95), PC-BOARD 11/E (Enhanced, support for 3 network notes, $179.95), and PC-BOARD 11/U (Unlimited, support for up to 99 network nodes, $999.95). PC-BOARD Version 11 features interactive scripts, comprehensive on-line help, baud rates from 300 to 9600, five different file transfer protocols, up to 99 public file directories and bulletins, and up to nine restricted access or public special conferences with their own menus and directories.

After a solid year's experience with PC-BOARD, I can recommend it to you wholeheartedly and without a single reser-

Introducing Periscope™ III

A new generation of debugging for the IBM PC, XT, AT and close compatibles

Now you can invest $995 and get the most powerful debugging tool available short of a $10,000 in-circuit emulator! The Periscope III board's hardware breakpoints and real-time trace buffer help you solve the really tough debugging problems. If you ever deal with errors in real-time systems, intermittent failures, interfacing with undocumented systems, or bottlenecks in your code, Periscope III may be just what you need!

Call TOLL-FREE 800/722-7006 for more information.

The PERISCOPE Company, Inc.

14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

vation. The documentation is clear and complete, the software is fast and robust, and the authors of PC-BOARD are extraordinarily helpful and responsive. PC-BOARD is constantly being enhanced with new features, and registered users are supplied with a password that allows them to download new releases directly from the author's "home" BBS. PC-BOARD would still be a steal at five times the price.

You can provide your customers with additional low-cost and low-stress means of reaching you for technical support by establishing CompuServe, Genie, Source, MCI Mail, and BIX accounts and publishing your EMAIL address on those services in your manuals. A high percentage of your users will already subscribe to one of these on-line services, and those who don't will get a little added motivational boost from your example. BIX is especially nice because it can be reached easily and at reasonable cost by European and Australian users through TYMNET gateways.

Updates

The last element of technical support that I want to discuss in this issue is updates. I feel very strongly that a software vendor owes it to his customers to provide low-cost and timely updates to program versions as they are released. A user should be rewarded, rather than punished, for buying an early version of a program.

The first tricky thing about updates is setting the price. I suggest that an update should be priced to just cover costs unless there has been such a dramatic increase in functionality that the user will perceive it as a new program rather than an improved program. You do need to examine both the direct and hidden costs very carefully, though, so that you don't end up "losing money on each one but making it up in volume." The direct costs are pretty easy to evaluate: printing, trimming, drilling and shrink-wrapping the manual; and the cost of the diskette, sleeve, packing peanuts, box, and mailing label. The hidden costs are a lot more challenging to get a handle on: processing the update order, printing an invoice, typing a mailing label, copying the distribution disk, boxing up the manual and disk and weighing it, filling out the UPS book, etc.

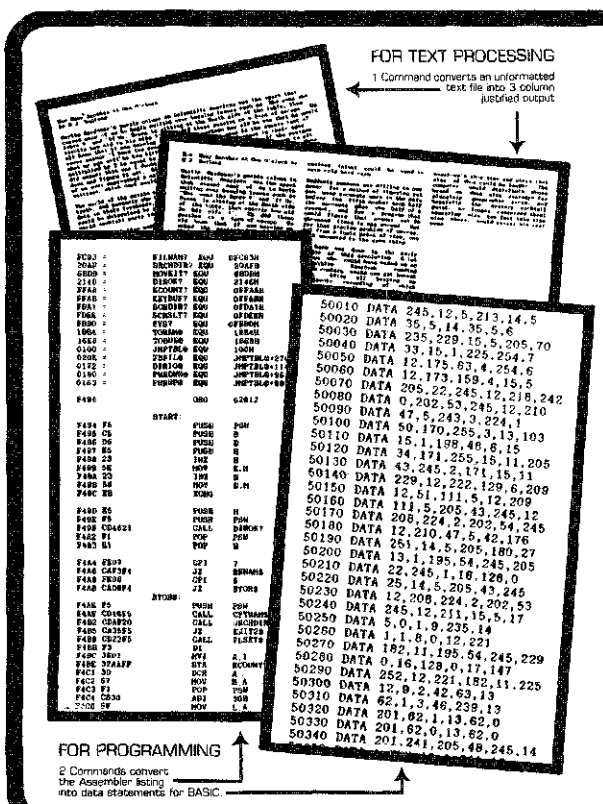
The other tricky aspect of updates is handling sales of the old version that occur right before the update is released. On the one hand, the user who buys Version 1 from you directly and two weeks later sees an advertisement for a vastly improved Version 2, is going to be justifiably resentful. It seems unfair to nick this guy for an update fee just because he didn't think to ask you on the phone whether there were going to be any major changes in the near future. On the other hand, you can't afford to bring your shipping to a standstill and cut off your cash flow for the month that precedes a new release.

One way to minimize the financial damage and frustration for all involved parties is to try to triage the orders into two groups. To the first group, individuals who are paying cash or by credit card for the software, ship the old version immediately with a postcard or letter that entitles them to a free update to the new software when it is ready. Orders from dealers, or from large companies on purchase orders, should simply be deferred if possible by advising them that an improved product is on the way and quoting them an appropriate delivery date.

Coming Up

In the next and final installment, we'll explore a grab bag of left-over subjects such as documentation and packaging, employees, payroll, insurance, and anything else *PJ* readers see fit to suggest. □

Ray Duncan is the president of Laboratory Microsystems, Inc., who sell Forth interpreters, compilers, and cross-compilers for a variety of microprocessors. LMI also does consulting work in Assembler, Forth, and C. In addition to running his software business, Ray Duncan frequently contributes articles to magazines and journals in the microcomputer field. His most recent book is the definitive, best-selling Advanced MS DOS, *published by Microsoft Press.*


FOR TEXT PROCESSING
1 Command converts an unformatted text file into 3 column justified output
FOR PROGRAMMING
2 Commands convert the Assembler listing into data statements for BASIC

CUT HOURS OFF TEXT PROCESSING TIME WITH:

TEXT-PRO

Use Text-Pro to make reusable, canned multiple Search & Replaces to use on different files.

Create Batch files, reformat, select, merge, split, sort files, insert records, do Global Search & Replace without going into the editor.

Convert variable to fixed length records & vice versa.
Produce documentation with real impact.
Format listings.

Makes light work of Code & Data Conversions—handles Code conversion problems with a single command.
Ideal for porting Code.

Chain several of Text-Pro's 29 programs together to produce dramatic changes in Text files.

Use to speed programming, conversion, text editing.

In use in major Electronic Publishing firms & Software Houses.

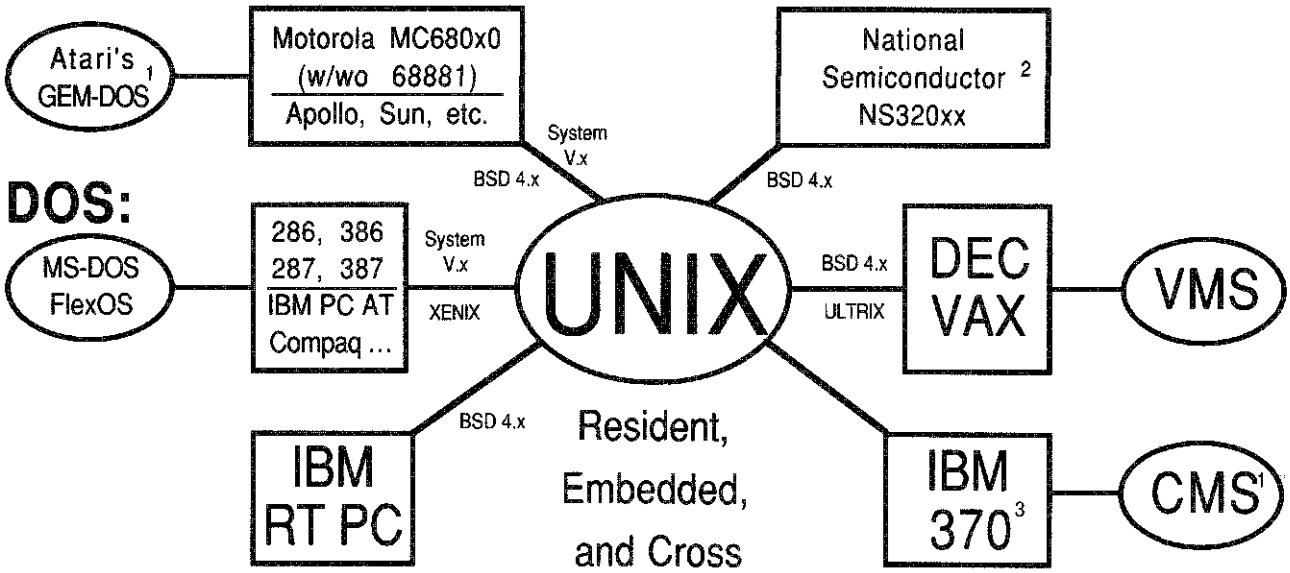
For MSDOS & PCDOS machines.

King Computer Services, Inc., 1016 N. New Hampshire
Los Angeles, CA 90029 (213) 661-2063

Number **91** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C and Pascal on:



We cut our teeth on UNIX, but have become famous on MS-DOS, which we enhanced with our UNIX-like **DOS Helper** ™ utilities: find (including tar), fgrep, cat, ls, mv, tail, uniq, and wc; and our superior optimizing compilers:

Professional Pascal ™ and **High C** ™ on the PC are now well-respected by organizations such as Ansa, Ashton-Tate, AutoDesk, Boeing (BCS), Daisy Systems Corp., Deloitte Haskins & Sells, Digital Research, GE, IBM, Lifetree, Migent, Multimate, NYU, Silvar-Lisco, Sky Computers, Symantec, Xerox/Ventura, ...and *Computer Language* magazine; *Dr. Dobbs' Journal; PC Tech Journal; PC Magazine;* the *Journal of Pascal, Ada, and Modula-2*...

We supply the **first, and still only,** compilers generating **32-bit protected-mode code for the 80386** under MS-DOS. And our newly upgraded MS-DOS real-mode compilers were used by Symantec for their Q&A™ product to exploit the power of the 80386 real-mode instruction set. (**Just released:** HC v1.4 and PP v2.7, May 1987.)

Our **C Validation Suite** will blow your C compiler out of the sea, while our C compiler tracks the emerging ANSI Standard and generates tighter code with far better lint-like feedback help than competing compilers.

And you'll love **Professional Pascal**'s Ada-like packages, true data abstraction, C-like bit manipulation, and much more, along with the tight code that is linkable with **High C**, or other C, object modules (and vice versa).

Our **Translator Writing System (TWS)** goes far beyond LEX and YACC, with fully automatic error recovery...

All uniformly implemented on UNIX, VMS, CMS, MS-DOS, FlexOS...

Professional software developers in need of **industrial-strength tools** should contact:

MetaWare™ INCORPORATED

MetaWare Incorporated
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060-4429
(408) 429-6382 (META)
Telex: 493-0879 (META UI)

The Clear Choice for Large Programming Projects.

Name _____
Company _____
Address _____
City, ST _____ Zip _____
Phone: (____) _____

PJ 07/87
Product:
Platform:

Circle what interests you:
PP HC TWS DOS Helper (DOS only
V.x 4.x DOS FlexOS VMS CMS
Sun Apollo Atari VAX 370
8086-family 80386pm 680x0 320xx

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others/owners: Ada/DoD; Apollo/Apollo; Atari/Atari; DEC,VAX,VMS/DEC; FlexOS,GEM-DOS/Digital Research Inc.; IBM,RT PC/IBM; MS-DOS/Microsoft; Q&A/Symantec; Sun/Sun; UNIX/AT&T.

Footnotes: 1. Atari, CMS versions available 7/87. 2. NS320xx version by special order. 3. UNIX not yet available on 370.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Unix Books, Part 3

by
Frank D. Greco

In this issue, Frank D. Greco concludes his survey of the literature on Unix.

Yates and Emerson
The Business Guide to the Unix System
Addison/Wesley

I'm not sure which market niche *The Business Guide to the Unix System* addresses. The book is not for programmers; it lacks technical information substantial or otherwise. Nor is it really a book for the business user because it is arguable whether business users of computers should use a raw operating system directly. That is the province of the computer programmer. I'm not saying that business people aren't smart enough; nothing could be further from the truth. To optimize their time and maximize their profits, though, businesses should use software that helps them do so. Using the Unix text processing tools (I call them documentation assembler languages) nroff and troff just does not cut it. Unix and its system tools allow *programmers* to quickly design and develop application and system programs for the benefit of end users. Business folk are best suited (no pun intended) for what they do best—business.

I really do not know what a "Unix System Management and Maintenance" chapter is doing in a "Business Guide." Having a business person use the tar, find, dump, and restor utilities is absurd. Should we ask accountants to write C programs next? Maybe lawyers should take shell programming classes and doctors become awk experts. Programmers should provide the interfaces to smooth the line-oriented inconsistencies and thereby increase the usability of these utilities.

I had to chuckle a little when I read in Chapter 4 of this book an attempt by the authors to convey to a *business* user in two small paragraphs the concept of an "inode," which is the Unix equivalent to PC DOS's File Control Block. A business user should study Unix internals before using the filesystem? I rest my case.

I believe the authors were trying to capitalize on the keywords "Unix" and "Business Guide" on the cover. I'm all for a catchy book title, but this one has serious conceptual problems.

Meets Intended Audience	?
Title Matches Contents	1
Accurate	7
Thorough	2
Use of Illustrations	9

Strong Points *Illustrations and choices of fonts were excellent*
Weak Points *The vi discussion is weak*
Comments *Would you buy a book titled* The Business Guide to Astrophysics*?*

R.C. Holt
Concurrent Euclid, the Unix System and Tunis
Addison-Wesley

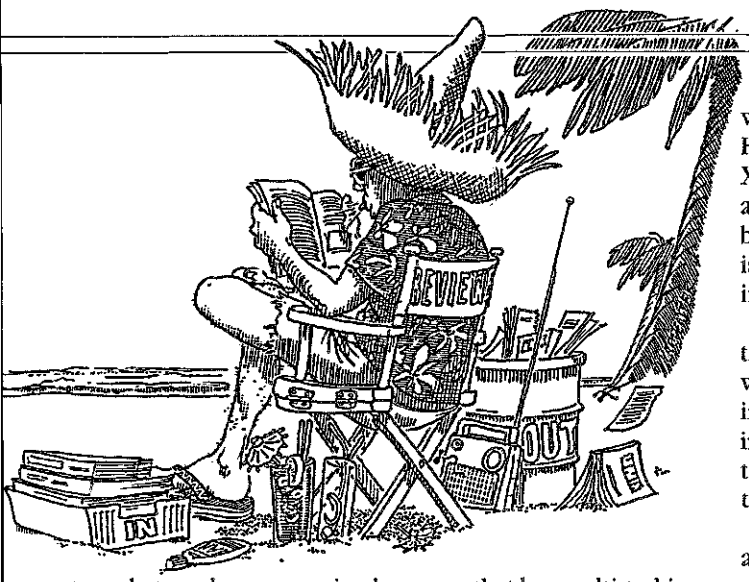
Written by a long time maven of concurrent programming, *Concurrent Euclid, the Unix System and Tunis* covers two multi-tasking operating systems (Unix and Tunis) and one multi-tasking programming language (Concurrent Euclid). The concept of concurrent or "parallel" processing is essential in the construction of operating systems, so Holt spends several chapters discussing fundamental principles. Besides being an expert on concurrency topics, Holt is equally an expert teacher and author. As in his excellent Pascal book, Holt precisely defines each new term or concept before he uses it, something that many new authors forget to do. The remainder of the book deals with in-depth discussions of the title topics.

The chapters on Unix discuss the user interface, file system, user processes, and the shell and provide a high level explanation of how the Unix kernel is implemented. The Unix chapters are well done except for a small nit: Holt covers what seems to be AT&T's Version 7 Unix. Most college students learning about the innards of Unix have access to a Berkeley version, typically BSD 4.1 or BSD 4.2. Although most of the AT&T and Berkeley concepts are the same or similar, many of the Berkeley enhancements will confuse students of operating systems.

For example, Holt properly tells the reader that directories under (Version 7) Unix are files that each contain an array of fixed 16-byte structures. The first 2 bytes represent the internal Unix name of a file (inode number or inumber), and the last 14 bytes represent what the user knows the file by (filename). On the other hand, Berkeley Unix's directories are variously sized structures; filenames can be up to 255 characters. A student following Holt's book and writing a system program that assumes 16-byte structs in a directory file will be in for a surprise when the program is run on BSD Unix. For the most part, though, if you want some additional information on how Unix does its thing, these chapters are enlightening. The discussion on the implementation of the Unix shell (Bourne shell) is particularly interesting.

Holt covers his favorite language and his favorite operating system, Concurrent Euclid and Tunis, respectively, in great detail. Comparisons of Tunis and Unix abound, and guess which OS always comes up superior. Tunis (Toronto UNIversity System) is an operating system written in a Pascal derivative, Concurrent Euclid. (I bet that most of you out there thought "Tunis" was another forced lexical derivative of "Unix"!) Tunis was developed to be an operating system similar to Unix but with improved performance and increased software "layering" for better maintenance. Tunis's implementation language, Concurrent Euclid, is a highly portable and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



strongly typed programming language that has multi-tasking primitives built right into the language itself. This "lightweight" process multi-tasking is similar to that of Concurrent Pascal, Modula, Ada, and the current non-protected mode version of Microsoft's Windows. The implementation of the Unix-compatible Tunis nucleus (kernel) is compared in detail with other historical influences (Multics, T.H.E., SUE/11, etc.). Overall, this is a very strong book for students studying operating systems.

Meets Intended Audience	9
Title Matches Contents	10
Accurate	8
Thorough	9
Use of Illustrations	1
Strong Points	*Excellent explanation of the shell, Discussion of software engineering and operating systems*
Weak Points	*No illustrations*
Comments	*Concurrency concepts superbly explained*

William B. Twitty
Unix on the IBM PC
Prentice-Hall

Approximately 1.5% of the Unix information in *Unix on the IBM PC* directly relates to the IBM PC; the remainder takes up Unix in general. Twitty talks about commands found in three of the most popular ports of Unix for the PC: PC/IX, Xenix, and Venix. The commands he describes are common to *all* Unix ports on *all* machines. For example, the Unix utility program "ls" is described as running under PC/IX, Xenix, and Venix. The ls utility also happens to run under VM/IX, 4.2 BSD Unix, HP-UX, System V Unix, Coherent, Idris, IS/3, FOS, ULTRIX, Zeus, Unisoft, Version 6, Version 7 Unix, System III Unix, etc., etc. So how is describing the ls command particularly helpful to a PC user?

And why should the author describe the cd, chmod, mv, cp, and other commands merely as available on all three versions? Why not tell the reader what to watch out for when using these commands on the PC specifically: What is an optimal directory size on the 10-megabyte XT using Venix? What is the effective usage of chmod on the single-user PC/IX? What are the nuances of the MS DOS cross development tools that come with Xenix? How does one use the PC/IX INed screen editor? How can the real-time facilities of Venix be used? What makes Xenix different from vanilla Unix? Sorry folks, these questions are not answered. This book is a book on Unix period! Don't be fooled by the title. Even though most of the information is correct, I felt cheated. A little nit: the old-fashioned spelling "disc"!

There are a few technical problems. One oversight is to tell the "vi" (a popular Unix based screen editor) user to determine whether vi knows about his or her particular terminal by typing the command "printenv." This is true *only* if you are using the C-shell (instead of the AT&T Bourne or Korn shell) from the Berkeley folks. The Bourne shell user should type "set" to see if the TERM variable is properly defined.

The system administration section, 25 or so pages, is probably the best part of this book. As a Unix book it rates an average grade; as a Unix-on-the-PC book, I give it a raised eyebrow.

Meets Intended Audience	7
Title Matches Contents	1
Accurate	7
Thorough	6
Use of Illustrations	1
Strong Points	*Comparison of three versions of Unix, System administration chapter*
Weak Points	*Minimal to no illustrations*
Comments	*Little PC-specific information, Not really applicable to some aspects of System V*

Kochan and Wood
Exploring The Unix System
Hayden

This fine book was written when both authors were employed by AT&T Bell Laboratories at Whippany, New Jersey. Currently they have their own consulting/training company called Pipeline Associates out of New York.

In the preface, Kochan and Wood announce that their main goal is ". . . primarily to teach the Unix philosophy. This philosophy preaches the doctrine that 'small is beautiful.' The Unix user is provided with a vast assortment of commands that perform small, well-defined functions and the tools necessary to combine these commands to perform more sophisticated functions." The authors continue, ". . . you will realize that this same principle forms the foundation of the structured programming discipline." I wonder if Unix creator Ken Thompson really thought about the "small is beautiful" principle consciously. Unix was created on a memory-limited processor, the segmented PDP-7. Since programs were constricted to 64K I & D space, there was no choice but to write small programs. In addition, the expensive PDP-7 disk drives were relatively fast, which may also have contributed to the many-small-utilities-on-disk way of thinking. Perhaps this age-old Unix doctrine was just born of necessity.

In the second chapter, Kochan and Wood say that the Unix operating system can be split into three groups: the scheduler, the file system, and the shell (I don't agree). A discussion of the Unix scheduler and swapper ensues. Recall that this is in the second chapter, a little early for these concepts to be

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

brought up; they are a little too technical. It's like having an 8:30 a.m. meeting without coffee. It may also install a subliminal block in newcomers to Unix. Why would a beginning user first care about the swapper and scheduler?

Kochan and Wood follow with, "Everyone on a Unix system has his own copy of the shell program." While this is logically correct, the shell is usually set up as a shared-text program. This means that there is only *one* physical copy of the shell in RAM and many users can share it. Any program for which you have the source or object code can be set up to be shared-text. An option to the Unix loader (ld -n) allows this.

The chapter on Getting Started is excellent beginning fare. Kochan and Wood walk the Unix neophyte through some of the fundamental notions such as directories, editing, and file management. The examples provided are perfect for helping the student digest the concepts. The authors don't sacrifice accuracy for ease of understanding. A minor misstatement occurs in a footnote: "On Berkeley Unix system 4.2, file names can be any length." While filenames under AT&T Unix can have up to 14 characters, as the authors say, Berkeley Unix, strictly speaking, allows up to 255 characters. This results from the unusual linked list in the directory file. Refer to my comments in the review of R.C. Holt's *Concurrent Euclid, The Unix System and Tunis*.

Kochan and Wood also do an excellent job in their descriptive chapter on the shell interface. Chapter 6 dedicates about 50 pages to shell command and shell programming examples. Shell variables, control structures, shell arithmetic, and parameter transmission are discussed as best they can be in one chapter. K&W also talk about something that most other Unix writers ignore or gloss over: shell quoting mechanisms. Part of any user's initiation into the Unix fraternity involves frustration and confusion with shell quoting. Several pages deal admirably with the topic.

The discussion of the echo utility/command does show the authors' AT&T Unix background. The \n that they imbed in a string presented to echo will *not* generate a newline under Berkeley Unix. Also the \c will *not* suppress echo's default newline under Berkeley Unix; a -n switch is necessary there.

The Program Development chapter is skimpy though Kochan and Wood apologize that they did not have enough space to go into detail on all the tools mentioned. They mention the Unix tool awk, the C language, sno, lex, yacc, adb, sccs, make, lint, prof and sdb. But that's all they do. The cursory glimpse even of make and sccs did not contribute much to this weak chapter.

Since Unix security is a controversial topic, I anticipated that Kochan and Wood, experts in security issues at Whippany NJ's Bell Laboratories, would discuss this topic in detail. Unfortunately, they don't. A scant discussion is devoted to user passwords, the standard Unix file permissions, and a token look at file encryption. After reviewing this book, I found out that Kochan and Wood have dedicated an entire book to the topic, *Unix System Security*.

The book closes with a decent discussion of Unix administration, with an all too brief (and rare) section on system accounting. For small systems this is not quite so important. On large systems (mainframes especially) it's very useful to know which and how users are using the machine's resources.

Kochan and Wood's *Exploring the Unix System* starts off with a bang, promising even more. Unfortunately, K&W run out of gas towards the end. Even though the fine chapter on shells and the decent look at Unix system administration compensate somewhat, that's not enough to keep the book's overall quality at the excellent level with which it begins.

Meets Intended Audience	6
Title Matches Contents	7
Accurate	9
Thorough	6
Use of Illustrations	8
Strong Points	*Shell chapter is excellent, Illustrations very clear*
Weak Points	*Program development tools not discussed well or not even discussed, Unix security slighted*
Comments	*Good overall book for shell users, but not for C programmers*

Birns, Brown, and Muster
Unix for People
Prentice-Hall

Inside this book you find the subtitle *A Modular Guide to the Unix Operating System: Visual Editing, Document Preparation, & Other Resources*. This is another misleading title (and subtitle) for a Unix book. Essentially, the authors cover vi, nroff, troff, and related information. The book should be retitled *Unix for Text Processor People*.

Basic Unix commands are covered towards the back of the book. For example, the fundamental cd command (Change Directory) is *first* covered on page 367.

Given the title of this book, you would assume that the instruction on using Unix would be from an end user point of view. Wrong, "/dev/null-breath," I am convinced that this book really fails at that. The chapter on vi is atrocious. How about this for a Unix-for-people (end-user) book: a "Truly Advanced Visual Editing" chapter that contains a section on "Cursor Relocation and Display Adjusting Commands." A book for an end user should talk in an end user's vocabulary, not in confusing "computerisms." This particular chapter happens to show the potential Unix end user how to create a keyboard macro in vi to input the lines:

```
Status Report,
  Mr. Spock

Yes, Captain
```

What a useful macro. I'm typing that one in all the time.

The only redeeming attribute of this book is that it covers the "eqn" and "tbl" pre-processors. However, the *Unix Primer* by Lomuto and Lomuto does a much better job. Avoid *Unix for People*—it's not. □

Meets Intended Audience	5
Title Matches Contents	1
Accurate	7
Thorough	2
Use of Illustrations	1
Strong Points	*Covers documentation tools*
Weak Points	*Fundamental Unix information delayed until the end*
Comments	*Minimal useful information for programmers, Not really applicable to some System V material*

Frank D. Greco is an independent consultant who specializes in highly usable computer systems. Readers may contact Frank at the following address:

Frank D. Greco
P.O. Box 470
Kearny, NJ 07032-470

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Complete Accounting Software with Source Code

Create Custom Vertical Accounting Modules

• General Ledger • Inventory
• Accounts Payable• Cost Accounting
• Accounts Receivable

Buy all five software modules for $200 or purchase them separately for $50 each. Each module comes complete with full documentation, an executable file and documented pretty-printed source code.The modules share an easy-to-use common user interface and work together to make accounting tasks simple. A low-cost OEM license is now available for software developers.

System requirements: Turbo® Pascal 3.0 or above, the Turbo Pascal Toolbox, and an IBM® PC or compatible. Each module is shipped on a DSDD 5 1/4" diskette. Shipping charges: $5 US, $10 Canada. Send check or money order to:

PS BUSINESS SYSTEMS

PS Business Systems
4400 Wallingford Ave N.
Suite 3
Seattle, WA 98103
(206) 632-4568

Number 82 on the Reader Service Card

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update In several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED**. including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE**-
  ASE is still only..... **$99**
  DEMO (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

TRANSPORTABLE ★ FULLY DOCUMENTED
EFFICIENT ★ AFFORDABLE

ASP includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

ASP is still only..... **$130**

ASE and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

ASPEN SYSTEMS

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number 86 on the Reader Service Card

Introducing

Quaid Analyzer

the tool that created CopyWrite

Now you can debug your own programs with a professional quality debugger - the one that unraveled every form of copy-protection used on the PC.

With the Quaid Analyzer, you can:

- ☐ See occurences of any interupt, with its meaning shown on the screen.
- ☐ View memory as text or instructions, scrolling as easily as you do with an editor.
- ☐ Run until a memory location or I/O port is changed.
- ☐ Protect your hard disk from accidental destruction.
- ☐ Analyze software without the source, even when it uses countermeasures to thwart tracing.
- ☐ See all stages of the boot load.

We kept the Quaid Analyzer off the market to avoid helping publishers with copy-protection. Now that copy-protection is gone, we can sell it to you.

The Quaid Analyzer is a software tool occupying 100K bytes. It runs on any IBM PC and most MS-DOS systems without hardware modification.

To order Call **(416) 961-8243**

Quaid Analyzer $99 U.S.

All orders shipped at our expense within a day. All major credit cards accepted.

or return coupon to:
45 Charles St. East
Third Floor, Dept. 603
Toronto, Ontario. M4Y 1S2

Payment method MC-Visa-Amex-Diners-Check
Card No. _____
Expiry Date _____
Name _____
Address _____
City/State _____
Phone No. _____
Signature _____

Quaid Software Limited

Ask about Disk Explorer the program that takes over where Quaid Analyzer leaves off.

Contact this Advertiser directly

REPRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Consultant Productivity and Effectiveness

by
Paul Barkley

Life is tough. Then you die.

Well, life sure *is* tough, and if you're an independent computer consultant, it might *really* be tough. Why? Because productivity is a killer.

How Much Time Do You "Waste"?

Let's say you're an employee of a corporation. You get paid a salary for working 40 hours a week. You might get only 30 hours of useful work done during that period. The rest of the time you might be wandering down to the water cooler, wandering into the rest room (to rest from your Herculean labors), taking a personal phone call or two, and socializing with the person in the next cubicle. That's just part of the work environment, and if someone asked you how much time you waste, you'd probably say about an hour a week. If you were surreptitiously monitored, though, it would probably come out to something like 10 hours. No one thinks he or she wastes very much time.

Monday morning of the following week you're suddenly an independent computer consultant. What do you think happens to your productive time? Does it increase? Nope—you wind up "wasting" even more time. The difference, though, is that you immediately *notice* this lost time because it directly impacts your financial well being. You notice that it seems to take forever to get anything done, and that even though you're very busy, you're not billing out much.

I've seen this phenomenon repeatedly with consultants just starting out. They can't believe how little gets accomplished each week with what seems like twice as much effort as before. They're working their tails off but billing out only a few hours a week. Something is dreadfully wrong.

I mentioned a figure of perhaps 10 hours per week in wasted activities. "Wasted" in the first case meant that your employer derived little or no benefit from these activities. Chit-chat doesn't translate well to the bottom line, and of course trips to the bathroom or the coffee maker are of similar (zero) corporate benefit. None of this directly benefits the independent either.

But there are worse things in store for an independent, things that drain your time like a jammed tap drains a cask of Ernest and Julio's finest. The culprits? Activities, particularly marketing and office administration, that are normally performed by other individuals in a corporation.

If you had an MBA in marketing and got a job as the assistant brand manager for a line of high fiber (that is, sawdust) breakfast food, you'd get paid to spend 40 hours a week planning marketing strategy, talking to buyers on the phone, meeting with advertising people, looking over print and electronic media campaigns, and in general figuring out how to sell more sawdust. ("Hey Frankie, Babe! You think we ought to put more sugar on the sawdust, or go with Nutri-Sweet? Let's take a meeting tomorrow with the ad guys and hash it out.") Swell job.

If you work in a corporate environment as an analyst or a programmer, you get paid for showing up every day and turning out deliverables. Marketing is somebody else's problem—probably another smart-off MBA's. In fact, there are a whole bunch of people around who go out and get contracts, who send bills to clients, who balance the corporate check book, who talk to pesky insurance and "financial planning" advisors who call up, and who take out the trash every night and vacuum your rug. Not *your* job, huh?

When you go independent, all these things *become* your job, and guess what? They don't pay diddley. You'll need to in-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

vest in a front end loader to cart off all the junk mail you'll get, and your phone will ring at least twice a day with people trying to sell you something you don't want (such as mutual funds for the cash you don't have). Okay, so that's one or two minutes per call, assuming you can instantly return to whatever it was that you were doing when the phone rang, and a half hour per day should be enough to process the routine mail and clean the office.

Add another hour a day for reading journals, and you'd better read very fast because you'll get about 4000 pages a month, only some of which will be crucial for you. If you index your periodicals, or otherwise make notes about the interesting stuff you read (not to mention trying to find shelf space to store all these magazines), you can plan on another couple hours a week.

If you're billing out some work, maybe an hour a week will take care of that, but if you itemize hours (as I do), it will take more in the neighborhood of two or three hours a week to log your hours and transfer them (manually or otherwise) to billing sheets.

Let's see, throw in an hour a week to pick up office supplies—try not to go more than once a week, even if office supplies include cookies and other essential snacks along with copy paper, tablets, postage stamps, notebooks, disk mailers, and writing instruments.

For every client you handle in a week, plan on one or two hours for loading their systems onto your hard disk, unloading them when you're done, keeping notes in their paper files about what you've done, and making suitable backups. Maybe you can bill this to your clients, but generally you eat this time along with the time you spend preparing their bills.

If any of your clients is out of town, add a half hour or more every time you prepare a disk to mail, including labeling it, putting it in a mailer, and typing an address label. Makes you wish you still had your secretary. If your clients are local, it's even worse. Plan on losing one or two hours per trip, minimum, even if you don't stay to exchange the funny jokes you heard last week.

Add a couple hours per week to pay your bills—rent (if you're in an office), your office supply bill, utilities, and so forth—and to record them in your bookkeeping system (you *do* have a bookkeeping system, I hope?). Add two or three hours a month to prepare your financial statements and ponder how well you're doing. Even if you turn this stuff over to an accountant, you'll still have to prepare the raw material, and with a good accounting program you might as well just finish the job yourself.

For me the big hidden killer is what I call "automation curiosity." A consultant is expected to be an expert in the field, and in the computer field that means you've got a considerable problem trying to stay abreast. I spent a year as a sales manager in a major computer store, and I learned that staying abreast was completely out of the question—all you can do is try not to drown as the Tsunamis roll over you.

Okay, let's forget the Apples and all that weird DEC stuff, and we'll leave out the Unix super micros, too. All we've got left are PCs and compatibles. Maybe it doesn't take that much effort to keep up with the boxes, even when a major announcement such as the PS/2 comes along (say 8 hours once every 5 years), but what about all the clones and the fast disk drives and the many graphics standards, not to mention laser printers, scanners, plotters, and mice? You could conceivably monitor hardware with only an hour or two of effort each week. The hard part is getting your hands on most of this stuff—as an independent you have to buy it out of your own pocket, find a client who has the stuff, or hang around a computer store where they don't want you taking the covers off and don't know anything about the product anyway.

They're working their tails off but billing out only a few hours a week.

The big problem is in software. I'm not talking just applications—keeping up with the latest releases of two or three word processors, Lotus, a couple data base managers, and your favorite communications software. You can plan on *at least* one revision per year for each of these products, and maybe a couple revisions on some. Each of these will take up one to eight hours or more of your time as you evaluate the changes and load them onto your system. Just hope that you don't have to convert your old file formats in order to use the new revision.

What about utilities? Everybody has his or her favorite utilities, and what interests somebody else may look just ridiculous to you. Still, you have to try a certain amount of this stuff, and it takes *time*—lots of time.

I hadn't been keeping up with hard disk utilities at all—must have been off in the ozone layer. Last week I returned to my system after working a couple weeks on someone else's new hardware for a special installation. When I got back on my old system, I thought my disk drive was broken. It took forever for Word Perfect to pull up directories, and the COPY command just moped along. Well, I thought, that's what you get for working on some really nice stuff—makes your own stuff look like junk.

The more I thought about it, though, the more I realized that my hard disk had been slowing down for a long time. After mentioning this to some of my colleagues (that's another couple hours a week, networking with your brethren or "sistren"), they all started telling me how they've been running disk optimization programs, packing fragmented files back together, freeing up contiguous disk space, and getting better response times. Another colleague pointed out that cleaning out the old dead directory entries helped (I'm not sure about that, but it sounds good), and a third told me horror tales about interleave factors that were set wrong for a given hard disk and controller combination and how he had a utility that would test interleave combinations and give you the optimal one.

Obviously, I needed to check all this out, and there went another four or five hours last week playing around with utilities (Norton and Mace and so forth—hey, I don't know *where* I've been for the last three years!). While I was at it, of course, I couldn't help trying out directory sorting and other things that I could have lived without but which were entertaining. Anything in the name of progress.

I log my hours all day long (yeah, it sounds pretty compulsive, I know, but it's mandatory), and I'm not going to tell you how many hours a week I spend on checking out new products or procedures. It's all in the log. Have you ever gotten a brilliant idea for a way to change some code that you've written and couldn't wait to get out of the shower to try it out? If you can't bill

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

that time, better chalk it off to programming experimentation, or automation curiosity. You wouldn't be in the business if you weren't inquisitive.

I touched on marketing at the beginning of this article. In general these activities include writing proposals; preparing competitive bids; preparing advertising or marketing materials such as brochures, stationary, cards, ads; sending unsolicited letters, resumes, or brochures to prospective clients; cold calling in person or on the phone; taking in-coming calls from prospective clients and following up with material sent out; and attending meetings of prospective clients.

Most technical people hate marketing for a variety of reasons, and they try to avoid it. Many technicians have abysmal marketing skills (and interestingly enough recognize this fact, a point in their favor). If you hate ties and the hypocrisy of dress, believe that everyone should be rational, and think small talk means discussing assembler routines, you probably aren't the marketing type. You not only don't have a business wardrobe, but you think only corporate sheep dress that way. You're certainly not going to waste your time trying to make a good impression on three-piece suited idiots at a stuffy company who just happen to have the hiring authority for consultants. The hell with them.

Well, some consultants seem to stay busy, even with this negative attitude about marketing, but everyone I've talked to who maintains this philosophy *and* stays busy seems to be tied up on long term contracts. The others with this attitude are starving.

Hmm. This suggests several things. First, it might be better to take a longer term job at a lower rate of pay to avoid all the marketing overhead and to be able to dress (and act) like a techie. You can probably get away with it and make some decent money, too. We're not going to pursue this line, though, because it strongly suggests that many people might be better off *not* being independent consultants, and that's not what this column is about. Keep it in mind, though. Consulting is not for everyone. (After you check out the financial analysis that follows, go back and compare it to long-term work, either on contract or as an employee.)

Secondly, it says that in order to be successful in consulting you're going to have to do something about all that unbillable time. Remember our traditional employee scenario when we got paid for 40 hours a week and spent 10 of it messing around?

If we could translate lost time directly from a traditional job to the independent life, we'd be able to bill 30 hours each week. But what about all those marketing chores, janitorial jobs, testing tasks, and general office work? We're talking *at least* another 10 hours a week. In fact, if we're not careful, by the time we get finished adding up all the unbillable time we'll be up to all 40 hours a week sucked into the chasm of unproductivity.

As a practical matter, if you're handling a variety of clients, as opposed to long term contracts, you can say goodbye to 20 hours a week. How are you going to survive under these conditions?

What Can You Do About It?

One method of survival ties into last issue's column ("Merger Mania," *PJ*, Vol. 5.3). You can grow large enough that these tasks are once again pushed off onto someone else. If you have three or four partners, a secretary can handle all the calls, the filing, and maybe the bookkeeping as well. You might even be able to hire yourself a face man with a nice haircut and a big smile to go around drumming up business.

That's begging the issue, though, by finding a way to avoid performing the unbillable tasks. You push them off on someone who will work for a lot less than you will. Then you can bill one hour and use that to pay for four hours of help.

If you've got to do all this yourself, though, there are several strategies that have to do with productivity and effectiveness—two separate issues. You simply have to get the most mileage out of your time.

You should be using an automated time billing system. You can either buy one for anywhere between a hundred and maybe eight hundred dollars, or you can write your own. Write your own if you're just starting out and have a lot of slack time. Use it to practice your analysis, design, and coding skills. *Finish it*, and then use it to make weekly or monthly billing as quick as possible. Don't keep modifying the system every week—it, too, will become a temporal black hole.

Your billing system should allow you to spend 5 minutes a day entering your time and at the end of the period spit out your invoices. You should be able to write bills up or down to adjust for unusual circumstances (I learned that from attorneys, but I never see them writing any work *down* unless someone complains loudly). Your billing system should also let you track totals on your clients and be tied into at least a crude measure of marketing effectiveness. For example, what percentage of your billable dollars last year came from add-on business from existing clients, what percentage from brand new clients, and what percentage grew out of each of these marketing activities: unsolicited proposals, listings in consultant directories, referrals from friends or social contacts, referrals from previous clients, etc.?

Learn how to dispose of the mail. Don't handle each piece fifteen times, putting it into piles to be processed later. Set aside some time for the mail and deal with each item as you open it. Put most of it in the trash. File (right then!) anything worth keeping. Once a year, at least, go through your files and throw out most of the stuff that you filed and never used.

Locate an indexing data base, such as Microsearch, instead of indexing all your own periodicals. Make notes of items of particular interest to you, but don't spend time indexing stuff that might not be useful. You never know what a client is going to want three months from now, and you can't index everything. If a client wants something, run it through the on-line data base. Once you locate the reference, you may find the periodical on your shelf.

Try to limit trips for supplies, phone calls from colleagues (hard to do because they're always so interesting), unnecessary trips to client sites, and so forth. Pretend that you're billing all this stuff to *yourself*, and think about writing yourself a check at the end of the week for this "unproductive" time. As the employee you wouldn't mind, but as the employer you'd be mad as hell about all this waste.

Having done everything you can to make yourself more efficient, what else can you do? Obviously, if you have to spend 20 hours a week on unbillable activities, that leaves only 20 hours of billable time if you want to work only 40 hours a week. Let's say you were earning $40,000 a year as an employee, including usable benefits. (Usable benefits are those that you'd pay for out of your own pocket if you had to, such as health and life insurance. Skip benefits that you didn't want and wouldn't replace because they're worth zero to you.) That's about $20 an hour. This means you must charge about $40 an hour just to break even in an independent practice. Given that you will have additional costs, charging only twice your previous rate will put you in the hole—you'd be better off as an employee.

You therefore must charge *more* than twice, say, three times, your previous rate, *or* you must plan on working more than 40 hours a week. If you want to work 60 hours a week, you can probably charge about $30 an hour (40 hours) and still make $40,000 a year after expenses, or you can bill 20

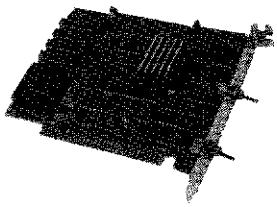
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MICROWAY ACCELERATES YOUR PC!

FastCACHE-286™

Runs your PC Faster than an AT!

Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz. Includes 8 kbytes of 55ns CACHE.



Compatible with Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics... **From $399**

LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles...**$549**

MegaPage with ϕK..... **$149**

MegaPage with 2 megabytes of HMOS RAM..... **$419**

MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. With 1 megabyte CMOS RAM..... **$699**

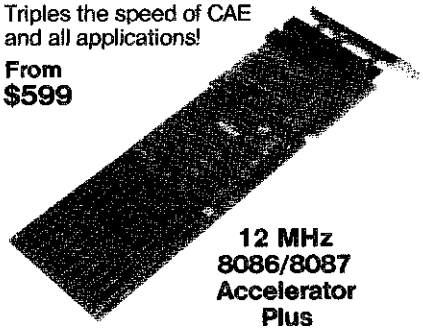
INTEL, JRAM, or Maynard ..... **CALL**

INTEL INBOARD 386 ϕK..... **$1325**

NUMBER SMASHER/ECM™

Triples the speed of CAE and all applications!

From $599



12 MHz 8086/8087 Accelerator Plus A Megabyte for DOS!

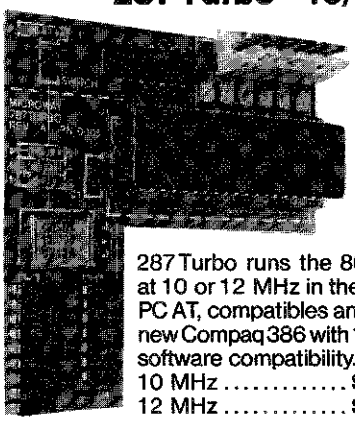
For the IBM PC, XT and compatibles

***PC Magazine* "Editor's Choice"**

8087 SOFTWARE

IBM BASIC COMPILER	$465
MICROSOFT QUICK BASIC	$79
87BASIC COMPILER PATCH	$150
87BASIC/INLINE	$200
IBM MACRO ASSEMBLER	$155
MS MACRO ASSEMBLER	$99
87MACRO/DEBUG	$199
MICROSOFT FORTRAN V4	$299
RM FORTRAN	$399
LAHEY FORTRAN F77L	$477
MS or LATTICE C	CALL
STSC APL★PLUS/PC	$450
STSC STATGRAPHICS	$675
SPSS/PC+	$695
87SFL Scientific Functions	$250
87FFT	$200
OBJ→ASM	$200
PHOENIX PRODUCTS	CALL

287 Turbo™-10/12



287 Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz **$450**

12 MHz **$550**

***PC Magazine* "Editor's Choice"**

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz..... **$105**
For the IBM PC, XT and compatibles

8087-2 8 MHz..... **$154**
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5 MHz..... **$179**
For the IBM PC AT and 286 compatibles

80287-6 6 MHz..... **$229**
For 8 MHz AT and compatibles

80287-8 8 MHz..... **$259**
For the 8 MHz 80286 accelerator cards and Compaq 386

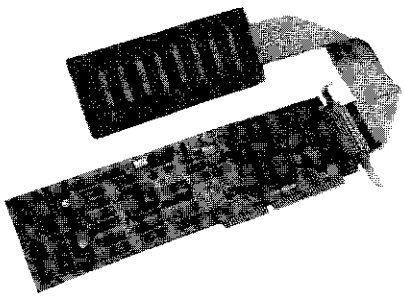
80287-10 10 MHz..... **$395**

PC-PAL™ Programmer..... **$395**

Call for great prices on V20, V30, 64K, 128K and 256K RAM

AT8™

Turns your AT into a high speed, multi-user Xenix business system!



8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA **$1299**

MICROWAY SOFTWARE FOR LOTUS 1-2-3™

FASTBREAK™ employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported..... **$79**

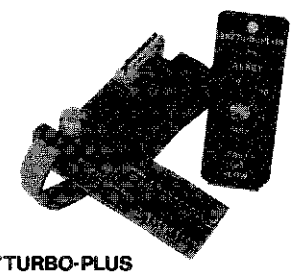
PowerDialer® Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used with DesqView **$79**

HOTLINK™ adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A.. **$99**

287 TURBO-PLUS™

Speeds up your AT

Adjustable 80286 Clock 6-12 MHz
10 MHz 80287 Clock
Plus Full Hardware Reset..... **$149**
Optional 80286-10..... **$175**



287TURBO-PLUS
With 80287 10 MHz..... **$549**
With 80287 12 MHz..... **$629**

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

MicroWay
P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

You Can Talk To Us!

MicroWay Europe
32 High Street
Kingston-Upon-Thames
Surrey England KT1 1HL
Telephone: 01-541-5466

Number 104 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

hours a week at $60 an hour and make the same amount. At the higher rate, of course, you're back to working only a 40 hour week.

We're all quantitatively oriented people here—you can set up your own spreadsheet to figure out where your break even point is. Just don't underestimate the amount of time lost to unbillable activities.

There's another fly in this ointment, though. These scenarios assumed that raising your rate to $60 an hour was reasonable (it may not be for your particular specialty and locale) and that you can get all the work you need under these conditions.

Let's say you settle for a regular 40 hour week—nothing shameful in that, is there? You track your time and find that if you're careful you can bill 20 hours of that. Now, the only way you're going to keep billing 20 hours a week, week after week, is to have more (much more) work than that because sooner or later you're going to pull a short week, where you finish what you're doing and nothing else materializes. Uh oh. In the last six years I've had three or four *months* when absolutely nothing came in (not back to back, fortunately). Drawing a totally unbillable week, at least in the past, was not at all surprising—happened all the time. They're not kidding in detective novels when the hero, working out of a crummy office, is glad to get a few days of billable work. Private detectives and consultants have a lot in common.

The only way you can consistently bill out at your target number of hours is to obtain more business, such that you've always got it piled up and waiting for you. How big a safety net do you need? One week's backlog? Two weeks'? A month's? I don't know many consultants who can get a month behind with their clients and still keep them interested. Clients tend to be impulsive and impatient. If you get too far behind, by the time you get to a project the client may well have changed his or her mind, or found someone else to do it, or completed the project in-house.

If you want a technical analysis of why you're going to have peaks and valleys and how you need a sizable backlog to prevent *any* off-weeks, grab a book on management science (or "operations research," as it's often called), and check out queuing models. Clients arrive (in your queue) somewhat randomly, so without a sizable backlog you're guaranteed, by probability theory, to have lulls.

One solution is to work more hours as your backlog grows. If there's a real danger of losing a project because it will take too long to get to it, you'd better put in some "overtime" cutting that backlog down to size. If you're backed up 80 hours (four weeks of 20 billable hours), you can cut that backlog by one week every week you work a total of 60 hours (40 billable, 20 unbillable). Four weeks of this effort, and your backlog will be gone. When this happens, if nothing else turns up, you can take off a week and you're still three weeks ahead of your planned 20 hour a week schedule. Not bad, as long as you don't mind the extra work. If you've never worked 60 hours a week in your life, you're kidding yourself if you think you're going to now.

The other thing that happens as your practice grows is that you feel an increasing pressure to hide out from all that unbillable time. Pleasurable activities, like perusing journals, suddenly become onerous chores—they're keeping you from getting other things done. You find yourself starting to read more quickly and becoming more selective about what you actually read and what you only scan. Your indexing activity drops off. You begin to ignore advances in your field (I already confessed to this). You don't initiate as many calls to colleagues as you should. You cut back on attending meetings, and the number of new prospects that you meet starts to dwindle.

This can have bad long term consequences and cause your business to dry up six to twelve months down the road. You may think you've got the world by the tail, but in fact you may well be reaping the benefit of all the hard marketing effort you put out last year. Clients don't show up by magic—something (besides your natural good looks and technological wizardry) drew them to your practice. Even word of mouth referrals need to be stoked by checking back with previous clients from time to time just to be pleasant and to remind them that you're still around. Takes work.

You have to find ways to cut out the wasted time. You not only have to make yourself more efficient, but you also have to make yourself more productive. I've found a number of strategies for this.

Besides automating your own office to cut down the bookkeeping chores, you should, in every way possible, automate your other efforts as well. If your clients trust you enough to have you design spiffy automated systems for *their* businesses, why can't you do the same for *your own* business?

Start with marketing activities. If you send out materials, try to make them generic (but still effective) enough that you can design them once and get a large number printed. Then send a pile out every week. Don't redesign the whole thing every month—you don't have the time. If you're making any money at all, hire professional designers and let them do their thing. The final product will be much better. If you don't have any graphic design skills at all, better find the money somewhere to hire the professionals or skip the direct mail entirely.

What about proposals? Figure out the best way to do a proposal. Depending on the kind of work that you go after, it may be appropriate to include a resume and a technical list of systems you've worked on. Put this together in the best way you know how and try to use it with as little change as possible, consistent with doing a good job. Write the very best boilerplate paragraphs that you can, and try to reuse them.

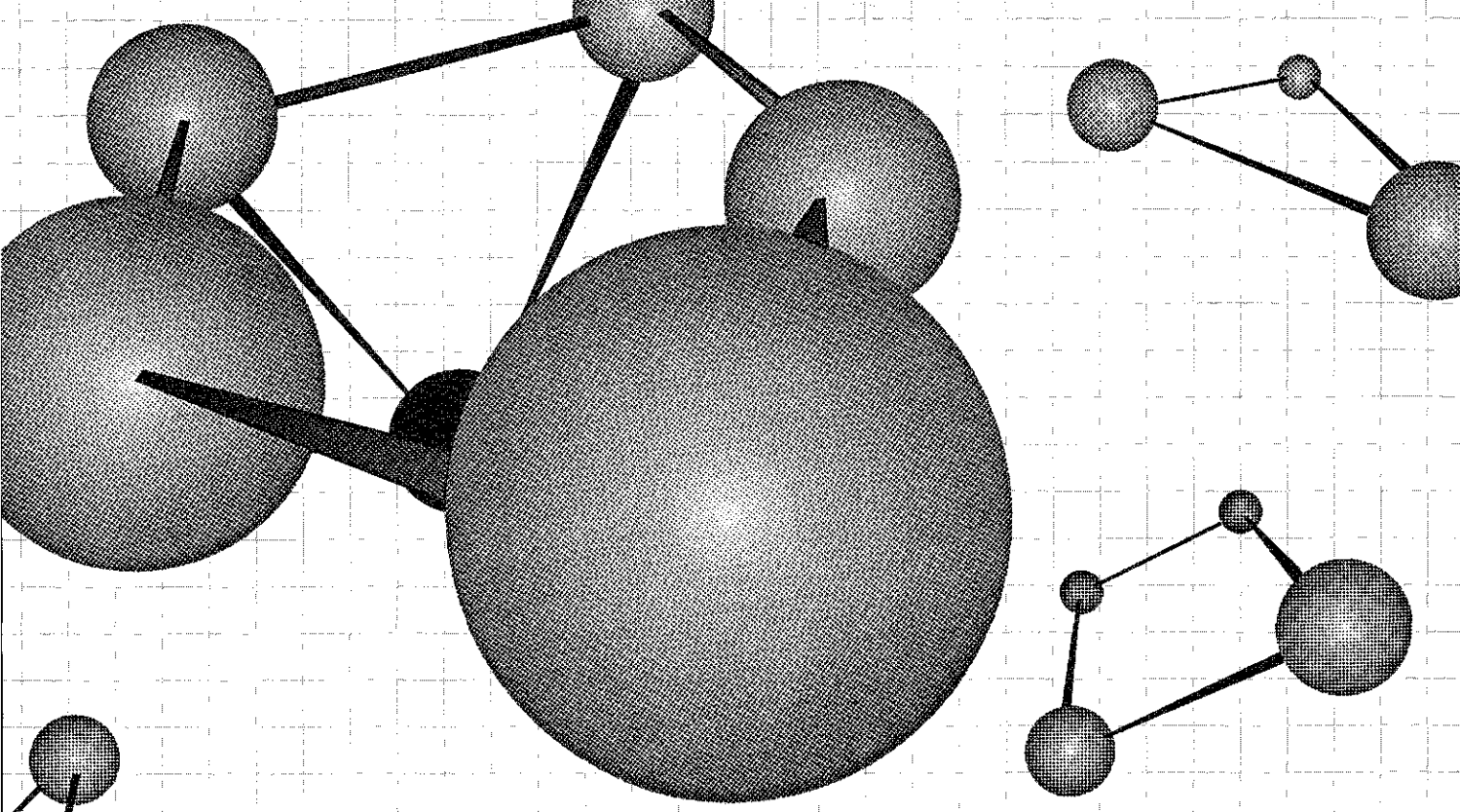
Become proficient with your word processor. If you can't type worth beans, for heaven's sake go out and get yourself a typing program (or a $5 typing book instead) and learn to type. Keyboarding skills are essential. As a consultant you may be able to get away with not playing golf, or with being a lousy dancer, and perhaps with being deathly afraid of spiders and snakes, but you must type reasonably well.

Draw up skeletons for contracts, too. Have your attorney draw up a boilerplate and explain where you can pretty safely do your own contracts. If your work is similar from client to client, you don't need to pay the attorney to pull the stuff off his word processor when you can use your own. Don't waste time drafting routine contracts from scratch.

Determine billing policies, both rates and terms, and stick to them where possible. Don't invent new policies every week—change them once or twice a year as conditions warrant.

If you develop code, try to either buy or develop your own application generator. It's far too expensive to code anything from scratch. At the very least, start building your tool box with modules you can string together with your editor or word processor. Do the best job you can on a particular module, say by writing the very cleanest multi-break report that you can, and then use that for a model for every one of your systems. You'll still have to code stuff by hand, but try to eliminate large blocks of boring coding. If you're on a fixed fee contract, this is absolutely essential. You can't fool around. If you're working on an hourly basis, this will reduce your hours, and thus your dollars billed, but it will greatly increase the quality of your work and make you very desirable to your

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Aztec C - the *best* C in the Micro-cosm

Why is Manx Aztec C the best C ?

Because Manx offers a combination of performance, features, and pricing that are unbeatable.

What kind of performance ? In the 2/87 issue of Computer Languages, our 3.2 release of Aztec C for PC-DOS turned out 109,489 iterations of the Dhrystone benchmark per minute to108,303 for Microsoft C. Our 4.1 code is even faster.

What kind of features ? Features like in-line assembly code, a professional make utility, library source, ROM support, debuggers, assemblers, and overlay linkers. Many of these features are unavailable or extremely high priced options in other systems. Microsoft, for instance, does not provide library source or in-line assembly.

What kind of pricing ? For $199, the Professional version of Aztec C gives you an Optimized C, a macro assembler, a librarian, overlay linker, extensive UNIX and MS-DOS specific library routines, a powerful debugger, sample programs, and over 500 pages of documentation. Manx Aztec C has a hard earned and well deserved reputation as a leading C software development system for PC-DOS and MS-DOS. At this price, it is also one of the best software values available.

Over 80 percent of our sales are repeat sales or are based on recommendations. That says it all. At Manx, we extend ourselves to the limit to provide the best software, the best service, and the best support possible. Call today, and join the over 50,000 users who know the Manx difference.

Manx also provides C systems for Macintosh, Amiga, CP/M-80, TRS-80, Apple //, 6502/65C02 ROM, 8080/Z80 ROM, 680x0 ROM, and 8086/80x86 ROM. **Call for prices and information.**

Aztec C86 4.1

PC-DOS • MS-DOS • ROM

Aztec C86-p Professional System $199

• optimized C with near, far, huge, small, and large memory • in-line assembler • in-line 8087/80287 • ANSI support • Fast Float (32 bit) • optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • MS-DOS 3.x file sharing & locking • comprehensive libraries of UNIX, MS-DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System.....$299

• includes all of Aztec C86-p • Unix utilities make, diff,grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System.....$499

• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 cross dev. option • One year of updates.

How To Become An Aztec C User

To become an Aztec C user call **800-221-0440.** From NJ or international locations call **201-542-2121. Telex: 4995812** or **FAX: 201-542-8386. C.O.D., VISA, Master Card, American Express, wire (domestic and international), and terms are available.**

Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee. Systems are upgradable by paying the difference in price plus $10. One and two day delivery is available for all domestic and most international destinations.

To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

REPRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Number 92 on the Reader Service Card

New Version 2.0

Complete C Programs in Half the Time, with *Instant-C*™

You can create programs much faster with Instant-C than with conventional programming tools. How? Because Instant-C is a high-performance interpreter, there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine*, **Editor's Choice** for best C interpreter.

Powerful **source-level debugging** saves your time. Conditional breakpoints, single-stepping by statement, source code backtraces, data monitoring, and many other debugging features make it easy to wipe out bugs quickly. Direct execution of any statement or function makes testing a breeze.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy."—*The C Journal*

Instant-C checks pointer references for reasonableness, and checks that array indexes are within declared bounds. This **run-time checking** stops your program as soon as errors occur, for easiest debugging.

Not only does Instant-C help you quickly change, test, check and debug your code, but it runs your program **fast enough for real-time** applications.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—*Computer Language*

Immediate feedback and precise diagnostics make Instant-C great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes Instant-C compatible with your existing programs.

Instant-C makes all parts of the programming task as fast as possible.

"Clearly, Instant-C is the performance champion."—*PC Tech Journal*

Version 2 works with MS-DOS and PC-DOS, and has a full 31 day **money back guarantee.** Instant-C is only $495. Order today! Call or write for full information.

Rational Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

Number **20** on the Reader Service Card

clients. They'll appreciate how much you can accomplish in a short period of time, and you can plan on coming back for more jobs. It's a dynamite competitive advantage, too.

The Hidden Benefits

This discussion has focused on efficiency, on ways to increase your productivity, but there's a big surprise hidden in here. There's another, more important, benefit of putting these measures in place. Recently, I heard Ray Youstra from IBM discuss issues related to information centers, and it occurred to me that the same principles apply to consultants as well. He pointed out that efficiency measures came out of studies done in factory environments earlier in this century. In a factory, you put in raw materials and factory labor and by combining them with some manufacturing process get a finished product out the other end. In an office, though, you put information and office labor in one end, operate on them with office procedures, and get *something* out the other end. What you get out is not nearly as evident as what comes out of a factory. (That's why a consulting contract must be careful to specify what the deliverables are—what's the tangible product?)

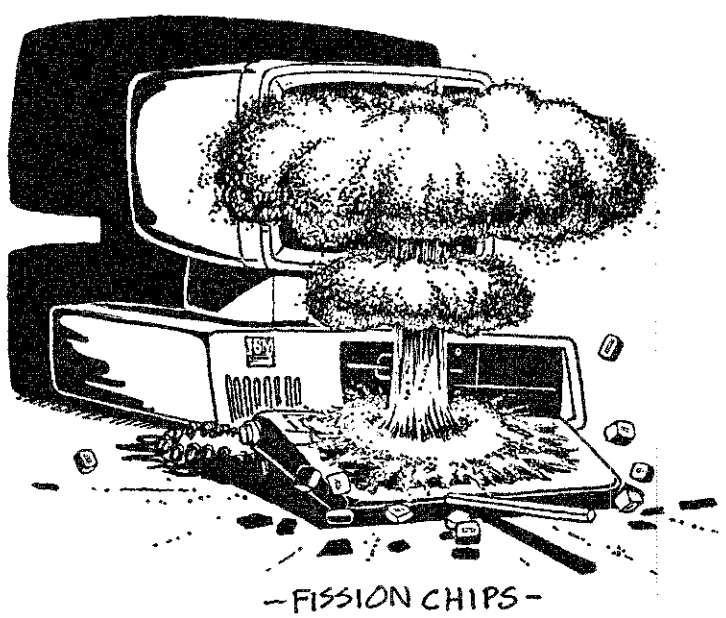
If you find ways for a factory worker to assemble more widgets per day, you'll make more money because more widgets will roll out the door. If you find ways for an office worker to type more lines of finished documents per day, you may not get any more (final) paper out the door. As an example, Mr. Youstra related an anecdote about an attorney's office where there was a large word processing pool. A visitor remarked that before automation they must have had an *enormous* typing pool. Nope. Seems that before automation they would send briefs through three drafts, and now briefs go through 40 revisions. They use the same number of people to produce the same number of briefs.

The big difference, though, is in effectiveness. Those briefs are a lot better than the older ones that weren't revised much, and they win more cases. The law firm is more effective even though they don't seem more efficient. It's a subtle point, but important. Strive to multiply your effectiveness.

So what happens when you begin automating your consulting practice? You achieve some efficiencies in some areas, some real productivity gains. More importantly, though, you greatly increase the quality of your work—remember the discussion about writing the very best report you can? As your quality goes up, using the same amount of your limited resources (your time), your effectiveness goes up. As your effectiveness goes up, you become more competitive, more likely to win business that would otherwise not be undertaken by the client at all, and in addition you can charge more for your services. Ahhhhh. The pride of turning out a better product, and the joy of making more money while doing it. Now that's an unbeatable combination! □

If you'd like to share your *productivity and effectiveness tips, drop a line to*

Paul Barkley
The Barbican Company
6845 Elm Street
Penthouse One
McLean, VA 22101



PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "Recommended. If you need a general purpose PC programming editor, look no further." His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated, "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195.** If you already own BRIEF, call for upgrade information.

TO ORDER CALL: 1-800-821-2492 (in MA call 617-337-6963)

As always, BRIEF comes with a 30 day, money back satisfaction guarantee.

Solution Systems™
541 Main Street
Suite 410
So. Weymouth, MA 02190
(617) 337-6963

Look at these BRIEF 2.0 enhancements!

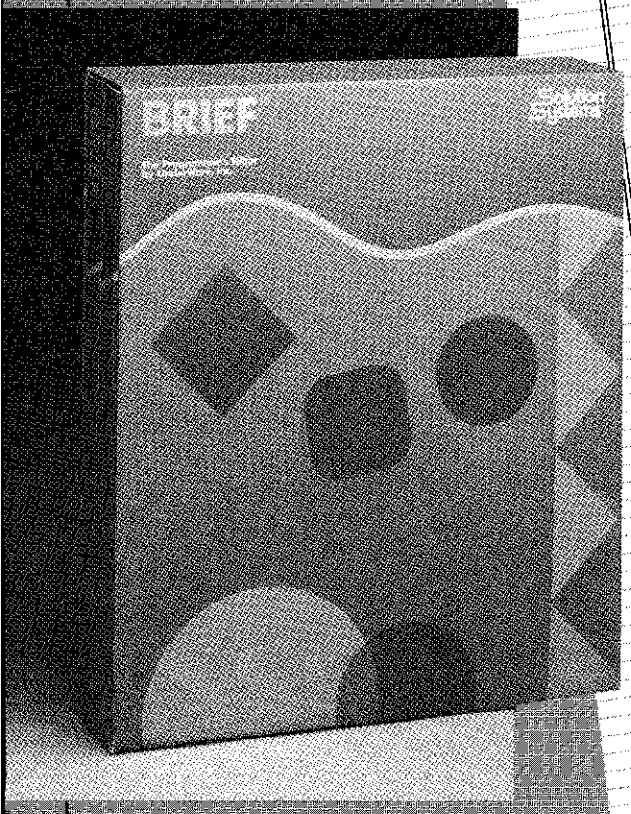
Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF SO popular!

Basic Features:

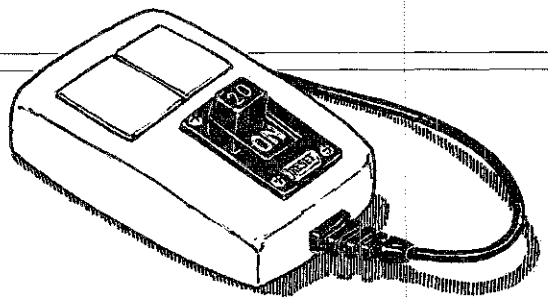
- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation



Requires an IBM PC or compatible with at least 192K RAM.
BRIEF is a trademark of UnderWare Inc.
Solution Systems is a trademark of Solution Systems.

Number 47 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



EGA Set/Reset Capabilities

by
Michael Abrash

This month I'll examine a powerful but confusing aspect of the EGA: the set/reset circuitry. I'll also discuss a few odds and ends and print reader responses to some of the questions I've posed in previous articles.

The Set/Reset Circuitry

At last we come to the final aspect of data flow through the Graphics Controller (GC) on write mode 0 writes, the set/reset circuitry. Figure 1 shows data flow on a write mode 0 write. The only difference between this figure and the similar figure in the last article is that on its way to each plane the rotated CPU data passes through the set/reset circuitry, which may or may not replace the CPU data with set/reset data. Briefly put, the set/reset circuitry enables the programmer to elect to independently replace the CPU data for any or all planes with either 00 or 0FFH.

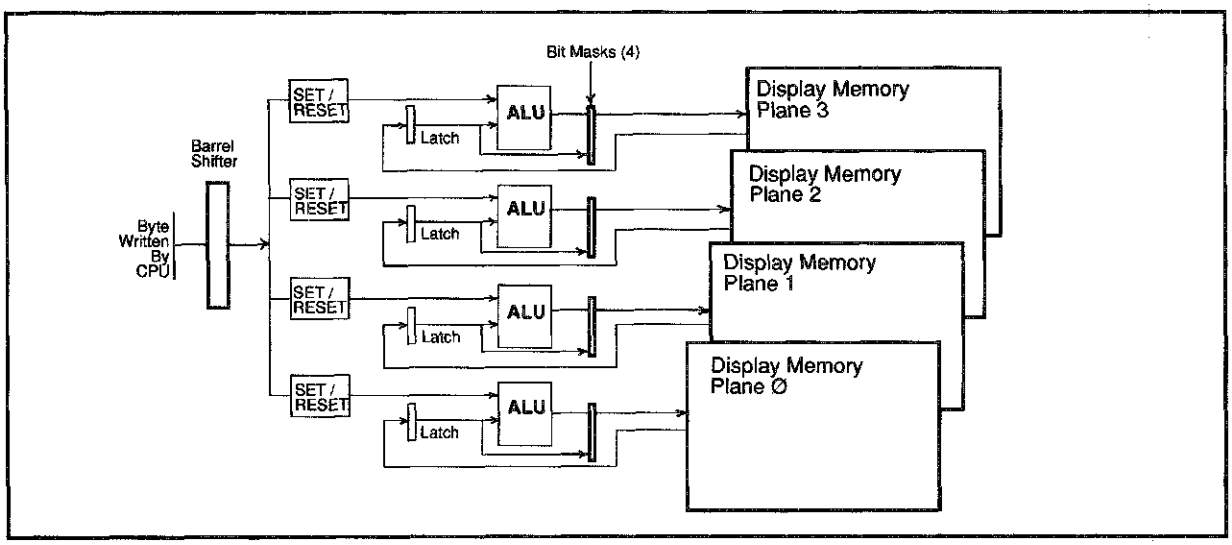


Figure 1: Data flow through the Graphics Controller chips on write mode

What is the use of such a feature? Well, the standard way to control color is to set the Map Mask register to enable writes to only those planes that need to be set to produce the desired color. For example, the Map Mask register would be set to 09H to draw in high-intensity blue; here, bits 0 and 3 are set to 1, so only the blue plane (plane 0) and the intensity plane (plane 3) are written to.

Remember, though, that planes that are disabled by the Map Mask register are not written to or modified in any way. This means that the approach above works only if the memory being written to is zeroed; if, however, the memory already contains non-zero data, that data will remain in the planes disabled by the Map Mask, and the end result will be that some planes contain the data just written and other planes contain old data. In short, color control using the Map Mask does not force all planes to contain the desired color; in particular, it is not possible to force some planes to zero and other planes to one in a single write with the Map Mask register.

The program in Listing 1 illustrates this problem. A green pattern (plane 1 set to 1, planes 0, 2 and 3 set to 0) is first written to display memory. Display memory is then filled with blue (only plane 0 set to 1), with a Map Mask setting of 01H. Where the blue crosses the green, cyan is produced, rather than blue, because the Map Mask register setting of 01H that produces blue leaves the green plane (plane 1) unchanged. In order to generate blue unconditionally, it would be necessary to set the Map Mask register to 0FH, clear memory, and then set the Map Mask register to 01H and fill with blue.

The set/reset circuitry can be used to force some planes to zero and others to one during a single write and so provides an efficient way to set all planes to a desired color. The set/reset circuitry works as follows:

For each of the bits 0 - 3 in the Enable Set/Reset register (Graphics Controller register 1) that is 1, the corresponding bit

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

in the Set/Reset register (GC register 0) is extended to a byte (0 or 0FFH) and replaces the CPU data for the corresponding plane. For each of the bits in the Enable Set/Reset register that is 0, the CPU data is used unchanged for that plane (normal operation). For example, if the Enable Set/Reset register is set to 01H and the Set/Reset register is set to 05H, the CPU data is replaced for plane 0 only (the blue plane), and the value it is replaced with is 0FFH (bit 0 of the Set/Reset register extended to a byte). Note that in this case, bits 1 - 3 of the Set/Reset register have no effect.

It is important to understand that the set/reset circuitry directly replaces CPU data in Graphics Controller data flow. Refer again to Figure 1 to see that the output of the set/reset circuitry passes through (and may be transformed by) the ALU and the bit mask before being written to memory, and even then the Map Mask register must enable the write. When using set/reset, it is generally desirable to set the Map Mask register to enable all planes the set/reset circuitry is controlling since those memory planes that are disabled by the Map Mask register cannot be modified, and the purpose of enabling set/reset for a plane is to force that plane to be set by the set/reset circuitry.

Listing 2 illustrates the use of set/reset to force a specific color to be written. This program is the same as that of Listing 1 except that set/reset rather than the Map Mask register is used to control color. The preexisting pattern is completely overwritten this time because the set/reset circuitry writes zeroes to planes that must be off as well as ones to planes that must be on.

Listing 3 illustrates the use of set/reset to control only some, rather than all, planes. Here, the set/reset circuitry forces plane 2 to 1 and planes 0 and 3 to 0. Because bit 1 of the Enable Set/Reset register is 0, however, set/reset does not affect plane 1; the CPU data goes unchanged to the plane 1 ALU. Consequently, the CPU data can be used to control the value written to plane 1. Given the settings of the other three planes, this means that each bit of CPU data that is 1 generates a brown pixel and each bit that is 0 generates a red pixel. Writing alternating bytes of 07H and 0E0H, then, creates a vertically striped pattern of brown and red.

In Listing 3, note that the vertical bars are 10 and 6 bytes wide and do not start on byte boundaries. Although set/reset replaces an entire byte of CPU data for a plane, the combination of set/reset for some planes and CPU data for other planes, as in the example above, can be used to control individual pixels.

There is no clearly defined role for the set/reset circuitry as there is for, say, the bit mask. In many cases, set/reset is largely interchangeable with CPU data, particularly with CPU data written in write mode 2 (write mode 2 operates similarly to the set/reset circuitry). The most powerful use of set/reset, in my experience, is in applications such as the example of Listing 3, where it is used to force the value written to certain planes while the CPU data is written to other planes.

Notes on Set/Reset

The set/reset circuitry is not active in write modes 1 or 2. An explanation of those two write modes is a topic for another article.

Be aware that because set/reset directly replaces CPU data, it does not necessarily have to force an entire display memory byte to 0 or 0FFH, even when set/reset is replacing CPU data for all planes. For example, if the Bit Mask register is set to 80H, the set/reset circuitry can modify only bit 7 of the destination byte in each plane since the other seven bits will come from the latches for each plane. Similarly, the set/reset value for each plane can be modified by that plane's ALU.

An Update

In an earlier article, I suggested the use of word OUTs (OUT DX,AX) to set indexed EGA registers. I have since been informed that this does not work in some early models of AT&T personal computers. If your software is likely to run in such a computer, I suggest you not use word OUTs. One good habit to get into is the use of macros or subroutine calls to set indexed registers; this practice makes it easy to convert from byte OUTs to word OUTs, or vice versa, if needed.

Reference Material

Several readers have asked where they can get information about programming the Color Graphics Adapter and Hercules Graphics Card. The *IBM Technical Reference, Options and Adapters* manual is a reference for the CGA and Monochrome Display Adapter, as well as the primary reference for the EGA. The best reference for the CGA, however, is "The IBM Color/Graphics Adapter" by Thomas V. Hoffmann, *PC Tech Journal*, Vol. 1, No. 1, July, 1983.

The primary reference for programming the HGC is the manual Hercules supplies with the board. Most companies that supply Hercules-compatible boards also provide programming information.

Reader Feedback

In my last article, I included responses from several readers explaining how to generate ROMable code at a specific address. This month's reader responses include answers to all three MASM questions I've posed and what surely is the definitive word on the problems of generating code at a specific address.

John Navas II answered all three MASM questions, as follows. The sample program to which he refers is shown in Listing 4.

> My solutions to your three puzzles range from crude to elegant to simple. I've enclosed a sample program that demonstrates all of them.
>
> (0) You are correct that MASM has only a single symbol space so that you cannot normally reuse symbols in different STRUCtures. However, if, for example, you are defining local STRUCtures for small procedures, there is a crude way to reuse symbols. Simply enclose an entire procedure including any local STRUCtures in a MACRO and declare all of the symbols LOCAL. When the macro is expanded, the assembler will generate unique symbols automatically, and there is no danger of referencing the wrong STRUCture. If you reuse the same MACRO name over and over, MASM shouldn't run out of memory. The downside is that the assembly takes longer, the listing is harder to read, and brackets must be used with LOCAL symbols instead of the more logical (.) STRUCture field-name operator (at least under MASM Version 4.0).
>
> (1) Using STRUCtures to define negative offsets from the BP register in a stack frame (typically for local dynamic storage allocation) requires the SIZE of the STRUCture to be subtracted from BP (for example, [bp-SIZE locs].field1).

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The EQU directive can make it all quite elegant. First EQUate a symbol to the SIZE of the STRUCture, adjusted if necessary to an even number. Then EQUate a second symbol to BP minus the first symbol, all enclosed in brackets. Subtract the first symbol from SP to allocate storage. Use the second symbol, followed by the STRUCture field name either separated by the (.) STRUCture field-name operator or enclosed in brackets, to access individual fields (for example, loc.w1 or loc[field1]).

(2) As you noted, the count for a DUP operator must be a constant expression. Since the location-counter is relocatable, it must be converted to a constant value before it can be used in a DUP count. The simplest way to convert it is to subtract the SEGMENT name. Your example might become:

```
DB      (2000h-($-segname)) DUP( 0 )
```

Note that the inner parentheses are necessary to force MASM to first perform the necessary conversion and that parentheses must of course enclose the DUP value.

Thank you, John. I must repeat a comment from the last article: The last-mentioned solution works only within the first module linked, which makes sense since the whole point of your solution is to generate an absolute value at assembly time and relocation takes place at link time.

Charles Clinton wrote at considerable length about the problem of putting code or data at an absolute offset, shedding light on the somewhat bewildering operation of MASM and proving both that there's a great deal for a patient and persistent reader to learn from the MASM manual and that there are mysteries that elude even the most patient and persistent reader.

Charles' letter follows. The letter was in the form of a letter to the editor; hence, the reference to the author in the third person. Please note that apart from correcting typos, I present the letter exactly as Charles wrote it; all editorial comments are his.

The difficulty that Michael Abrash is encountering in attempting to convince MASM to assemble code up to a certain location (*PJ* 5.1, p. 42) is that MASM has no idea where the code that it is currently assembling is going to end up. The code is assembled into the current program segment, by which MASM means "a collection of instructions and/or data whose address are all relative to the same segment register" (MASM 4.0 Ref. Man., Sect. 3.4, p. 27) and not relative to an absolute memory address or offset.

The solution to the problem involves making the assumption that since we know where the code is going to go, we can talk about absolute memory locations by describing them relative to the code.

The section on arithmetic operators (Sect. 5.3.1, p. 78) says that all arithmetic expressions must either be integers or, in some special cases, "relocatable memory operands." The section on relocatable [memory] operands (Sect. 5.2.3, p. 69) defines them as "any symbol that represents the memory address. . . of an instruction or of data. . . and [has] no explicit value until the program has been linked." Examining the definition of the current location counter (Sect. 5.2.4, p. 69), we see that it has "the same attributes" as a "near label." Flipping around some more, we find the section on symbol declarations (Sect. 4.4, p. 54) that implies that a label generates a symbol. This means that the $ is a relocatable memory operand (if our assumptions have not gone awry).

Going back to the section on arithmetic operators, we find that we can add or subtract a constant from a relocatable memory operand or we can find the absolute difference between two memory operands in the same [assembler] segment. We can exploit the difference operation to find our location in memory. If we know that a particular label is going to be at a particular location in memory and we know how to calculate the distance between two labels, we can calculate the location of a second label.

For example, with the problem that Mr. Abrash is presenting we need to determine the location of the end of the assembled code so that we can zero fill to the end of the ROM chip. If we define a label at the start of the ROM chip, like this:

```
code        segment     'ROM'
            assume      cs: code
            org         0
  rom_start:
            ... code ...
```

we can then calculate how many bytes have been assembled, thusly:

```
number_assbld = $ - offset rom_start
```

The number of bytes that need to be filled is the length of the ROM chip (2000H) less the number of bytes assembled:

```
to_fill = 2000H - number_assbld
```

The statement that does the filling is then:

```
    db      to_fill dup (0)
```

Putting it all together, and simplifying the expressions, yields:

```
code            segment     'ROM'
                assume      cs: code
                org         0
rom_start:
        ... code ...
                db          2000H - ($ - offset rom_start) dup (0)
code    ends
        end
```

The key to this solution is remembering that MASM will maintain that it does not know in what manner the linker will manipulate the segment being assembled. In fact, in most cases the linker will concatenate many code and data segments and then concatenate the segments into groups (which are not necessarily contiguous) and classes. This flexibility and power is what gives one the ability to mix multiple languages and memory models in a single program and allow exact specification of the ordering and arrangement of memory. Unfortunately, it has the side effect of making some of the simplest tasks complicated.

The seemingly obvious solution to this problem of using the "at address" combine clause in the segment declaration will not work as Microsoft's intended use of this was to allow the user to describe existing data areas, and not define them. So they did not make the expression evaluator smart enough to recognize that the current location counter ("$") can sometimes represent a fixed address.

A subtler problem seems to exist in the offset operator. At first glance, it would seem that

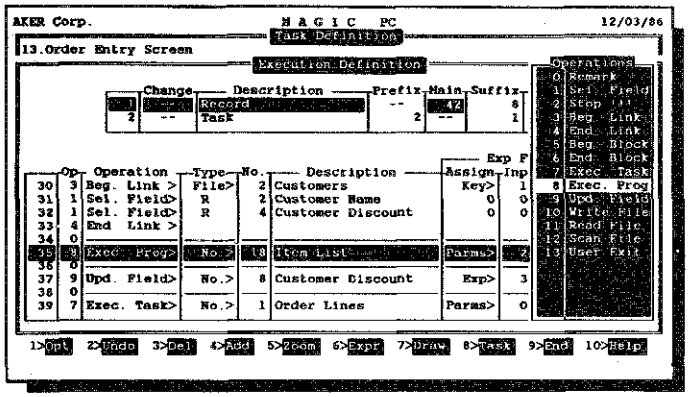
```
    db      2000h - offset $ dup (0)
```

should produce the desired effect. Although the MASM manual claims that the offset operator returns the number of bytes between the symbol and the beginning of its segment (Sect. 5.3.13, p. 88), a little experimenting shows that this is not entirely true. The .TYPE operand (Sect. 5.3.15, p. 89) reveals some curious things about the way MASM represents numbers. Assembling the following code reveals the problem:

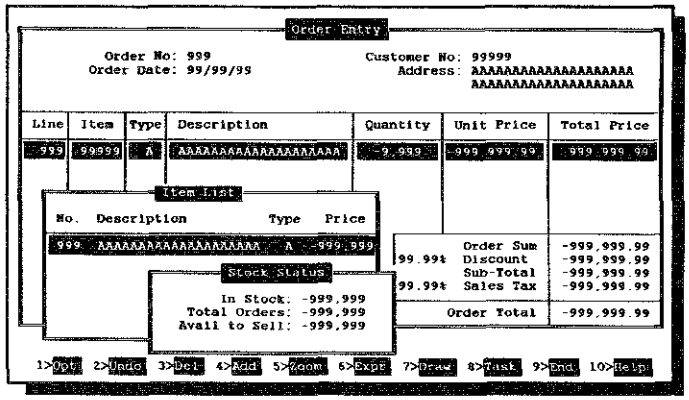
```
    code        segment     'ROM'
                assume      cs: code
                org         10
    a           =           10
    atype       =           .type a
    b           =           offset $
    btype       =           .type b
    code        ends
                end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Develop DB applications 10 times faster without the coding pain...you'll swear it's Magic



Visual Programming with Magic PC looks as simple as this. To design an application you quickly place your design specifications into menu-driven Task Tables without having to write a single line of code. For example, just by highlighting the Execute Program operation on the left screen and also highlighting the Item List program in the Program Menu, you tell Magic PC to Zoom into the Item List program through the window shown on the right screen. The window will automatically scroll the Item List data horizontally and vertically, and allow query, "cut and paste" copy or even creation of new Items.



Free yourself from coding

Database professionals throughout the world are discovering a new way to dramatically cut development time.

So can you! With **Magic PC, the Visual Database Language** by Aker.

Consultants, VAR's, Software Houses and DP MIS professionals: If you develop DB applications for a living, now you can tackle any database application 10 times faster than with your DBMS or 4GL.

What makes you so fast with Magic PC? It's not magic...it's simply because Magic PC finally frees you from coding. And doesn't coding take up most of your time right now?

Magic PC lets you leverage your design skills instead of wasting your time coding. Now you can generate a fully functional prototype in just hours for quick customer feedback, and easily refine the same prototype to a finished application.

All you do is enter your system design specifications directly into Magic PC's non-procedural menu-driven Tables, as ideas come to mind, and Magic PC generates the programs for you automatically.

Magic PC gives you a free hand to design powerful data management systems limited only by your own imagination. Without the time consuming mechanical details of conventional procedural programming. There's your competitive edge. The rest is up to you.

Your biggest time saving comes from Magic PC's dynamic adaptation to spontaneous design changes. You're free to change your design on the fly, and Magic PC automatically updates your programs and data files online. No more time wasted maintaining each program manually with every small change.

Visual Programming Power

You program with Magic PC by describing your data elements with **Data Dictionary Tables** (Files, Fields, Keys), and placing your system design specifications into **Task Description Tables.**

The Tasks can be nested within one another or dynamically Link to satellite Tasks, to give you true One-to-Many relational database power.

Only 13 Task **Operations** harness the power of Magic PC. Operations are specific enough to eliminate the need for tiresome coding, yet elastic enough to produce robust custom applications.

Use the Task building blocks to quickly generate Online Programs: Screens, Window Zooms, Menus; or Batch Programs: Reports, Updates, Data Import/Export and much more.

You develop the Task Tables visually on the screen by highlighting selections from Window Zooms and pop-up menu-driven Tables. You're not forced to follow any particular Table sequence, and there's no coding to slow you down. It's that simple.

You can apply mathematical and logical Expressions, or use the built-in Functions directly in the Task Tables to automate conditional Task processing, to display custom error messages or even invoke external applications such as spreadsheet, word processing or communication programs, transparently from within your Magic PC application programs.

Magic PC generates your application by fusing all your Data Dictionary and Program Tables seamlessly into a single **Integrated Library,** and automatically maintains changes online for optimal, bug-free performance, so you always get it right the first time.

Your application is executed at runtime by a **Magic Run** engine for stand-alone operation, and you can distribute your applications at a low cost and protect your design. Magic Run has a built-in visual interface to manipulate data and get on-the-spot ad-hoc information without any commands or syntax, simply by highlighting selections from menus. Data validation, security and error-checking are done automatically for you without programming.

Magic PC has built-in support for File and Record Locking so you can design multi-user applications for a local area network, and share data with any number of Magic Run users.

Magic PC integrates the Btrieve file manager internally, supporting the B-Tree file structure for fast high performance data access, and fault tolerant recovery during power failures.

Magic PC's powerful **Window Zoom** lets you design composite screens with windows to probe deep into the application through nested windows and manipulate the data underneath. By Zooming from any field, your end-user can conveniently query, copy or even create data in other programs directly through the windows, without stopping their screen session. The window frame size does not limit the available Data View since each window has built-in horizontal and vertical scrolling.

Magic PC is the professional's choice:

IBM France: "IBM encourages Magic PC and salutes such evolution..."

Israeli Air Force: "We were convinced that it was not possible to have a design tool powerful enough to implement real life applications without a programming language. Magic PC changed our mind..."

PC Magazine: "If the thought of programming database applications makes you tremble, Magic PC is for you. The applications generator saves users from the need to deal with much dreaded computer language code..."

PC Tech Journal: "Magic PC is probably the best integrated database application and screen generator that we have seen...very smooth system, and smoothness comes at a premium these days..."

PC World: "Relational data managers and application generators that offer power without programming are a bit like perpetual motion machines — very rare. Into that vacuum comes Magic PC, a data management tool without language that is ideal for turnkey applications..."

PC Week: "Rather than use a written programming language the user is given a great deal of freedom and power to create complex relational database applications...this package is a true time-saver..."

Get your Magic Tutorial for only $19.95

See for yourself how fast and powerful visual programming can be for you. Order your copy of **Magic PC Tutorial** including Tutorial disks, and a step-by-step tutorial to quickly set up an Order Entry application without coding. All for only $19.95 and We'll credit the Tutorial cost towards the $695 Magic PC price for up to 30 days.

Or save $500 at no risk!

Yes, for a short time only, qualified VAR's can save almost $500 off the list price and get the complete unprotected **Magic PC** software and documentation at a **special VAR price of $199*** with no risk. Keep it at this low price only if it's as good as we say, or return it within 30 days for a full refund if you're not completely satisfied. Act now and you'll also get 2 Magic Run (runtime modules) for only $95.

Order now for immediate delivery

Call this toll-free number now with your credit card or COD charge, or send the Order Coupon below today with your check to Aker.

1-800-345-MAGIC
in CA 714-250-1718

VISA MasterCard AMERICAN EXPRESS

Yes, please rush me the following
Prices include 2nd day shipping

☐ Magic PC Tutorial	$ 25.95
☐ Magic PC (VAR's only)	$209.00
☐ Magic PC	$705.00
☐ Magic Run	$100.00
in CA add 6% tax	$_____
Total	$_____

Ship to: _____
Address: _____
City/ST/Zip: _____
Phone: _____

OEM inquiries are welcome. Prices valid in North America only.
*Less $19.95 restocking fee. Limit one per customer, subject to availability. Not for resale.

System requirements: IBM PC, XT, AT and 100% compatibles, PC-DOS 2.0 or later, 512K and hard disk.

AKER

Aker Corp. 18007 Skypark Cir. B2, Irvine, CA 92714
Tlx 4931184 AKER UI
Aker S.A. 11 Route de Florissant CH-1206, Geneve
Switzerland Tlx 421792 AKER CH

Trademarks: Magic PC, The Visual Database Language, Window Zoom, Magic Run, Magic LAN and Magic PC Tutorial are trademarks of Aker Corp., IBM PC and PC-DOS are trademarks of IBM Corp., Novell is a trademark of Novell Inc., Btrieve is a trademark of Softcraft Inc.

Number **116** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Tools

For Programmers & Non-Programmers

Get 'State of the Art' performance and save valuable time with these high quality utilities!

Opt-Tech Sort™

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can even read the data. Example: 1,000 records of 80 bytes can be read, sorted and a new file written in less than 10 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or called as a subroutine to over 25 different programming languages.

All the sorting, record selection and reformatting facilities you need are included. A partial list of features includes: The ability to process files of any size. Numerous filetypes are supported including Sequential, Random, Delimited, Btrieve, dBASE II & III and many others. Up to 10 key fields can be specified (ascending or descending order). Over 16 different types of data supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes.

MS-DOS $149. ★ NEW ★ Xenix $249.

★ NEW ★ VERSION **On-Line Help**™ ★ NEW ★ VERSION

On-Line Help allows you to easily add "Help Windows" to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable Help Systems.

The resident version allows you to add help to any system. Your Help System is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire.

The callable version allows you to easily display help windows from your programs. A simple call to the help system makes the window appear. The original screen is automatically restored when the help window is cleared. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location.

MS-DOS $149. **Demo $10.** (apply toward purchase).

Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit your previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. It's a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted.

To order or to receive additional information just call and receive immediate highly qualified attention!

Opt-Tech Data Processing
P.O. Box 678 — Zephyr Cove, NV 89448
(702) 588-3737

Number **16** on the Reader Service Card

Upon examining the listing, we see:

```
                Name                    Type            Value     Attr

A . . . . . .                           Number          000A
ATYPE . . . . . .                       Number          0020

B . . . . . .                           Number          000A      CODE
BTYPE . . . . . .                       Number          0022
```

By the MASM definition of OFFSET, we would expect A and B to have identical values and both be plain numbers. Rather, what we see is that OFFSET is failing to strip off the attribute of 'CODE' from the expression "OFFSET $." Attributes are a little tricky in MASM as they are not defined in the MASM manual. They are not in the index, and the only mention of attributes that I can find are in the sections "Location-Counter Operand" (Sect. 5.2.4, p. 69) and ".TYPE Operator" (Sect. 5.3.15, p. 89), both of which I consider cryptic descriptions at best. There does not seem to be an operator for changing the attribute of an expression, which is what is called for here.

Thank you, Mr. Clinton, for guiding us deep into the murky depths of MASM as it pertains to the problem of generating code or data at an absolute offset. As has been my experience with MASM, there is indeed an explanation and a workaround, but it sure seems harder than need be to perform what should be a straightforward task. □

Code follows.

EVERLOCK COPY PROTECTION

Designed for user-transparency, clone compatibility & strength. It features:

- no I/O plugs or special media
- FULL hard disk & cartridge support
- file-server network support
- variable number of installs (0-99)
- demo diskette option with unlock
- protected upgrades by modem/BBS

All this for **$495** with no meter counts.
Free info & demo disk.
(Duplication services also available).

AZ-Tech Software, Inc.
305 E. Franklin, #A4A
Richmond, MO 64085
(816) 776-8153

Number **51** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code from Abrash, EGA Set/Reset Capabilities

1

```
; Listing 1.
; Program to illustrate operation of Map Mask register when drawing
;  to memory that already contains data.
; Assembled with MASM 4.0, linked with LINK 3.51.
; By Michael Abrash, 4/26/87.
;
stack   segment para stack 'STACK'
        db      512 dup(?)
stack   ends
;
EGA_VIDEO_SEGMENT       equ     0a000h  ;EGA display memory segment
;
; EGA register equates.
;
TS_INDEX        equ     3c4h    ;TS index register
TS_MAP_MASK     equ     2       ;TS map mask register
;
; Macro to set indexed register INDEX of TS chip to SETTING.
;
SETTS   macro   INDEX, SETTING
        mov     dx,TS_INDEX
        mov     al,INDEX
        out     dx,al
        inc     dx
        mov     al,SETTING
        out     dx,al
        dec     dx
        endm
;
cseg    segment para public 'CODE'
        assume  cs:cseg
start   proc    near
;
; Select 640x350 graphics mode.
;
        mov     ax,010h
        int     10h
;
        mov     ax,EGA_VIDEO_SEGMENT
        mov     es,ax                   ;point to video memory
;
; Draw 18 10-scan-line high horizontal bars in green, 10 scan lines apart.
;
        SETTS   TS_MAP_MASK,02h         ;map mask setting enables only
                                        ; plane 1, the green plane
        sub     di,di           ;start at beginning of video memory
        mov     al,0ffh
        mov     bp,18           ;# bars to draw
HorzBarLoop:
        mov     cx,80*10        ;# bytes per horizontal bar
        rep stosb               ;draw bar
        add     di,80*10        ;point to start of next bar
        dec     bp
        jnz     HorzBarLoop
;
; Fill screen with blue, using Map Mask register to enable writes
; to blue plane only.
;
        SETTS   TS_MAP_MASK,01h         ;map mask setting enables only
                                        ; plane 0, the blue plane
        sub     di,di
        mov     cx,80*350               ;# bytes per screen
        mov     al,0ffh
        rep stosb                       ;perform fill (affects only
                                        ; plane 0, the blue plane)
;
; Exit to DOS.
;
        mov     ah,4ch
        int     21h
start   endp
cseg    ends
        end     start
```

2

```
; Listing 2.
; Program to illustrate operation of set/reset circuitry to force
;  setting of memory that already contains data.
; Assembled with MASM 4.0, linked with LINK 3.51.
; By Michael Abrash, 4/26/87.
;
stack   segment para stack 'STACK'
        db      512 dup(?)
stack   ends
;
EGA_VIDEO_SEGMENT       equ     0a000h  ;EGA display memory segment
;
; EGA register equates.
;
TS_INDEX        equ     3c4h    ;TS index register
TS_MAP_MASK     equ     2       ;TS map mask register
GC_INDEX        equ     3ceh    ;GC index register
GC_SET_RESET    equ     0       ;GC set/reset register
GC_ENABLE_SET_RESET equ 1       ;GC enable set/reset register
;
; Macro to set indexed register INDEX of TS chip to SETTING.
;
SETTS   macro   INDEX, SETTING
        mov     dx,TS_INDEX
        mov     al,INDEX
        out     dx,al
        inc     dx
        mov     al,SETTING
        out     dx,al
        dec     dx
        endm
;
; Macro to set indexed register INDEX of GC chip to SETTING.
;
SETGC   macro   INDEX, SETTING
        mov     dx,GC_INDEX
        mov     al,INDEX
        out     dx,al
        inc     dx
        mov     al,SETTING
        out     dx,al
        dec     dx
        endm
;
cseg    segment para public 'CODE'
        assume  cs:cseg
start   proc    near
;
; Select 640x350 graphics mode.
;
        mov     ax,010h
        int     10h
;
        mov     ax,EGA_VIDEO_SEGMENT
        mov     es,ax                   ;point to video memory
;
; Draw 18 10-scan-line high horizontal bars in green, 10 scan lines apart.
;
        SETTS   TS_MAP_MASK,02h         ;map mask setting enables only
                                        ; plane 1, the green plane
        sub     di,di           ;start at beginning of video memory
        mov     al,0ffh
        mov     bp,18           ;# bars to draw
HorzBarLoop:
        mov     cx,80*10        ;# bytes per horizontal bar
        rep stosb               ;draw bar
        add     di,80*10        ;point to start of next bar
        dec     bp
        jnz     HorzBarLoop
;
; Fill screen with blue, using set/reset to force plane 0 to 1's and all
; other plane to 0's.
;
        SETTS   TS_MAP_MASK,0fh         ;must set map mask to enable all
                                        ; planes, so set/reset values can
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
                                        ; be written to memory
        SETGC   GC_ENABLE_SET_RESET,0fh ;CPU data to all planes will be
                                        ; replaced by set/reset value
        SETGC   GC_SET_RESET,01h        ;set/reset value is 0ffh for plane 0
                                        ; (the blue plane) and 0 for other
                                        ; planes
        sub     di,di
        mov     cx,80*350               ;# bytes per screen
        mov     al,0ffh                 ;since set/reset is enabled for all
                                        ; planes, the CPU data is ignored-
                                        ; only the act of writing is
                                        ; important
        rep stosb                       ;perform fill (affects all planes)
;
; Turn off set/reset.
;
        SETGC   GC_ENABLE_SET_RESET,0
;
; Exit to DOS.
;
        mov     ah,4ch
        int     21h
start   endp
cseg    ends
        end     start
```

```
; Listing 3.
; Program to illustrate operation of set/reset circuitry in conjunction
;  with CPU data to modify setting of memory that already contains data.
; Assembled with MASM 4.0, linked with LINK 3.51.
; By Michael Abrash, 4/26/87.
;
stack   segment para stack 'STACK'
        db      512 dup(?)
stack   ends
;
EGA_VIDEO_SEGMENT       equ     0a000h  ;EGA display memory segment
;
; EGA register equates.
;
TS_INDEX        equ     3c4h    ;TS index register
TS_MAP_MASK     equ     2       ;TS map mask register
GC_INDEX        equ     3ceh    ;GC index register
GC_SET_RESET    equ     0       ;GC set/reset register
GC_ENABLE_SET_RESET equ 1       ;GC enable set/reset register
;
; Macro to set indexed register INDEX of TS chip to SETTING.
;
SETTS   macro   INDEX, SETTING
        mov     dx,TS_INDEX
        mov     al,INDEX
        out     dx,al
        inc     dx
        mov     al,SETTING
        out     dx,al
        dec     dx
        endm
;
; Macro to set indexed register INDEX of GC chip to SETTING.
;
SETGC   macro   INDEX, SETTING
        mov     dx,GC_INDEX
        mov     al,INDEX
```



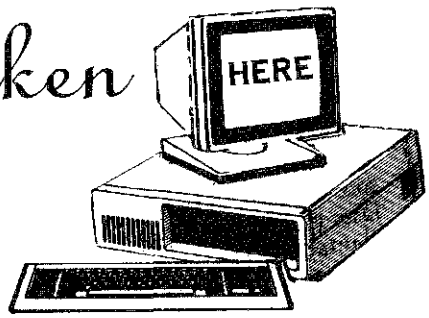
```
        out     dx,al
        inc     dx
        mov     al,SETTING
        out     dx,al
        dec     dx
        endm
;
cseg    segment para public 'CODE'
        assume  cs:cseg
start   proc    near
;
; Select 640x350 graphics mode.
;
        mov     ax,010h
        int     10h
;
        mov     ax,EGA_VIDEO_SEGMENT
        mov     es,ax                   ;point to video memory
;
; Draw 18 10-scan-line high horizontal bars in green, 10 scan lines apart.
;
        SETTS   TS_MAP_MASK,02h         ;map mask setting enables only
                                        ; plane 1, the green plane
        sub     di,di           ;start at beginning of video memory
        mov     al,0ffh
        mov     bp,18           ;# bars to draw
HorzBarLoop:
        mov     cx,80*10        ;# bytes per horizontal bar
        rep stosb               ;draw bar
        add     di,80*10        ;point to start of next bar
        dec     bp
        jnz     HorzBarLoop
;
; Fill screen with alternating bars of red and brown, using CPU data
; to set plane 1 and set/reset to set planes 0, 2 & 3.
;
        SETTS   TS_MAP_MASK,0fh         ;must set map mask to enable all
                                        ; planes, so set/reset values can
                                        ; be written to planes 0, 2 & 3
                                        ; and CPU data can be written to
                                        ; plane 1 (the green plane)
        SETGC   GC_ENABLE_SET_RESET,0dh ;CPU data to planes 0, 2 & 3 will be
                                        ; replaced by set/reset value
        SETGC   GC_SET_RESET,04h        ;set/reset value is 0ffh for plane 2
                                        ; (the red plane) and 0 for other
                                        ; planes
        sub     di,di
        mov     cx,80*350/2             ;# words per screen
        mov     ax,07e0h                ;CPU data controls only plane 1;
                                        ; set/reset controls other planes
        rep stosw                       ;perform fill (affects all planes)
;
; Turn off set/reset.
;
        SETGC   GC_ENABLE_SET_RESET,0
;
; Exit to DOS.
;
        mov     ah,4ch
        int     21h
start   endp
cseg    ends
        end     start
```


PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EGA Set/Reset Capabilities, Continued — 5

```
; Listing 4.
; By John Navas.
cseg    SEGMENT
        ASSUME  cs:cseg
mproc   MACRO                           ; for local names
        LOCAL   locs,field1,field2,locl,loc
locs    STRUC                           ; local storage on stack
field1  DW      ?
field2  DW      ?
locs    ENDS
locl    EQU     SIZE locs + 1 AND -2    ; even length
loc     EQU     [bp-locl]               ; negative offset from bp
sub     PROC
        push    bp                      ; save old bp
        mov     bp,sp                   ; setup new bp
        sub     sp,locl                 ; allocate local storage
        mov     loc[field1],1           ; access local storage
        mov     loc[field2],2
        mov     sp,bp                   ; deallocate storage
        pop     bp                      ; restore old bp
        ret                             ; return to caller
sub     ENDP
        ENDM                            ; mproc may be reused
        mproc                           ; expand local names
        DB      2000h - ( $ - cseg ) DUP( 0 )
        call    sub                     ; forced by DB to 2000h
cseg    ENDS
        END
```


dBASE spoken



If you do any dBASE consulting or applications work, you have something new to talk about. And to talk to. ELLIE, the most powerful natural language system in the world—bar none—has just become available as a memory resident interface to dBASE III/III Plus.

Memory-resident? Not for me...

Don't bet on it. This memory resident "utility" comes up from the dot prompt, calmly accepts your questions, and chugs away for 10-120 seconds. Then, press a key, and dBASE does the rest—executes the resulting code and puts the answer in front of your nose.

Sure. List customer for balance < 0. *Right?*

If you like. Or: *Show me the workers in Boston that haven't yet taken our advanced marketing course, sorting them by department and salaries.* From a 5-file relational database. Think about coding that for a minute. Now think about coding it in a minute. ELLIE can.

See our ad in July *Business Software*. As a professional courtesy to *PJ* readers, clip this ad and take $50 off the advertised price—till August 1 only. That makes your cost $139 (single workstation).

Your users will be talking about ELLIE, soon. You should start getting to know her today.

512K minimum RAM, hard drive recommended. We're happy to answer your questions, at (212) 316-9078. NY residents add applicable sales tax. Courtesy rate not honored without ELFSOFT logo; no facsimiles. dBASE III and dBASE III Plus are trademarks of Ashton-Tate Corporation.

ELFSOFT
210 W 101 NY NY 10025

Number **113** on the Reader Service Card

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

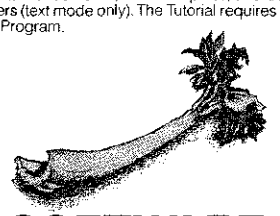
NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!
1-800-CALL-800 x8088

VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.



SOFTWARE GARDEN, INC.
Dept. J
P.O. Box 373, Newton Highlands, MA 02161

Number **108** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Proposed ANSI C Language Standard

by
Rex Jaeschke

© *Rex Jaeschke, 1987. All rights reserved.*

The const and volatile Keywords

Two of the new keywords added by ANSI C are **const** and **volatile**. Both are type modifiers and may be specified in type declarations along with any other type and with each other. The syntax of **const** and **volatile** is identical although their meanings are quite different.

const and the Programmer

An identifier declared as **const** cannot be used as an lvalue. It can, however, be given an initial value via an initialization list. For example,

```
const int total = 100;
static const int minimum;
const char *keywords[] = {"help", "add",
        "delete", "change", "display"};
```

Here, the **auto** variable **total** is initialized to 100, but thereafter, its value may not be changed. Any attempt to modify a **const** identifier generates a syntax error at compile-time. The variable **minimum** is initialized to zero since no initializer list is specified and it has class static.

The primary value of **const** is to ensure, at compile-time, that certain locations are never stored into. This is particularly important for those of you generating ROMable code. Without **const** protection, any attempt to write into read-only memory at run-time will generate some sort of fatal exception and the only solution is to find the bug and reburn the ROM, a messy and (possibly) expensive task. With **const** support, a compiler is able to detect these problems. **const** code can also reliably be placed in code (or text) sections and can be placed in shared memory.

const and the Optimizer

The other main value of **const** is that it potentially helps the compiler generate more optimum code, particularly across function calls. According to one optimizer authority on the ANSI C committee, Gary Jeter,

> The **const** modifier lessens the worst-case assumptions that must be made at function calls whenever the formal parameters are declared as pointer to const object. Since the only way to set the value of a const variable is through initialization, the definition of a const variable represents a constant propagation opportunity for the optimizer. In the compilations where const variables are declared but not defined, the optimizer can treat these globals as if they were literals in nearly all optimizations except constant folding. Hence, there are more common subexpressions, code motion, and strength reduction candidates.

const and Function Prototypes

As discussed in Part 6 in this series (*PJ*, Vol. 5.2), the **const** keyword may be used in function prototypes. Examples from **stdio.h** and **string.h** are

```
int remove(const char *pathname);
int rename(const char *old, const char *new);
FILE *fopen(const char *pathname, const char *type);
void setbuf(FILE *stream, char *buf);

char *strcpy(char *s1, const char *s2);
char *strcat(char *s1, const char *s2);
```

This mechanism is used to inform the optimizer about possible optimization opportunities across function calls as discussed above.

The volatile Modifier

The **volatile** modifier is used to specify that a variable may be used and/or updated asynchronously to the program's execution. Therefore, the compiler had better not optimize away seemingly redundant code as follows:

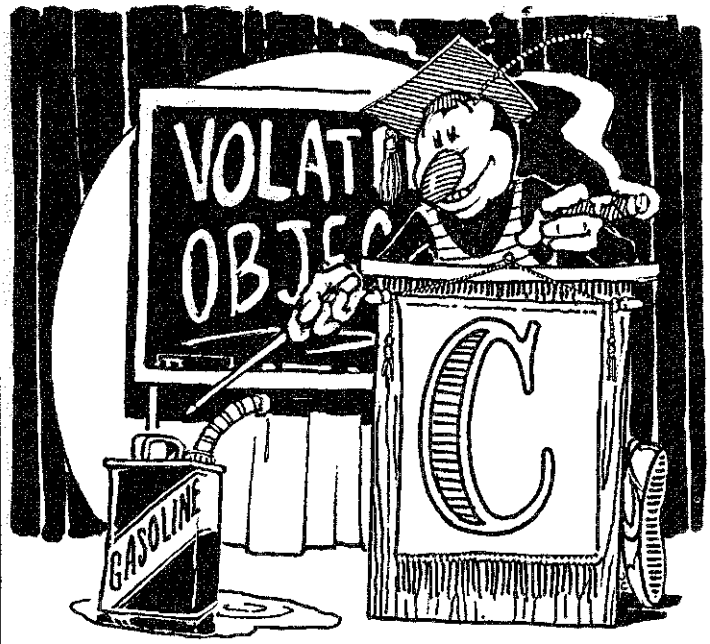
```
volatile int *pvi;

*pvi = 4;
*pvi = 4;
```

To the uninitiated, the last statement looks to be superfluous; however, if the pointer is mapped into some device register or I/O port, we may be pulsing some device four times, twice. If we specify **volatile**, the compiler knows to exclude this object from certain optimizations. Without the equivalent of volatile, your only recourse, typically, is to disable all optimization for this compilation, something not necessarily desirable or possible with some compilers. (Note that the new preprocessor directive **#pragma** provides a way for compilers to enable and disable optimization on a line-by-line basis.)

When dealing with **volatile** objects, the compiler is also obliged to update them immediately and not to wait until the next sequence point (such as a semicolon, ||, &&, ?:, etc.) since some other process may be waiting on or constantly inspecting their values.

Interrupt handlers and device drivers often use shared memory to keep track of events, so they are prime candidates to have **volatile** data objects.

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Default Typing and Aggregates

If these keywords are specified alone, type **int** is assumed. For example, the declarations

```
volatile int vi;
volatile vi;
```

are equivalent. The keywords may also be used with structures, unions, and enumerations and in typedefs.

```
const struct complex {
        double real;
        double imag;
} cs;

struct complex c;
```

Here, **cs** is a structure of type **complex** that has the **const** attribute. Note that the **const** attribute applies to the definition of **cs**, not to the structure tag **complex**. Therefore, **c** has class **auto** or is a global depending on its location in a source file.

const or **volatile** can be made to "stick" to a type if you use **typedef**.

```
typedef const int const_int;

const_int ci1;
const_int ci2;
```

Now **ci1** and **ci2** are both **const ints** and no **const** modifier is needed or allowed.

It is possible for only part of an aggregate to have **const** or **volatile** attributes as follows:

```
struct device_status {
        const int attributes;
        int interrupt_count;
        volatile int status;
        volatile char io_register;
};
```

Combining const and volatile

The type modifiers may be combined in any order:

```
const volatile int i;
volatile const int i;
```

Both declarations define an integer **i** that cannot be used as an lvalue and is probably updated/inspected asynchronously by some other process. The **const** attribute merely stops the programmer from modifying **i**—it does not stop other programs, so its value may still change—just not as a result of any code which declares it to be **const**.

const, volatile and Pointers

The type modifiers make for some interesting properties when applied to pointers. Four groups of objects can now be pointed to:

1. Normal pointers, pointing to non-**const**, non-**volatile** objects. For example,

```
char *pc;
```

2. Pointers to **const** objects:

```
const char *pcc;
```

3. Pointers to **volatile** objects:

```
volatile char *pvc;
```

4. Pointers to **const**, **volatile** objects:

```
const volatile char *pcvc;
```

In all cases, the object being pointed to is a **char**; however, the attributes of that **char** vary. These differences are quite important since the compiler is obliged to handle **const** and **volatile** in a consistent manner. When a **const** or **volatile** variable is assigned to another variable, do the **const** and **volatile** attributes go with it, and if so, how does the compiler police this?

Normal pointers can be assigned to pointers which point to **const**, **volatile**, or **const** and **volatile** objects without error. The reverse is not true without an explicit cast and, even if done in a syntactically correct manner, may result in undefined behavior; that is, the **const** and **volatile** attributes partake in the decision as to whether the arguments across an assignment operator are assignment-compatible.

A particularly interesting twist is that when used with pointers, **const** and **volatile** may appear twice in the same declaration. For example,

```
int *pi;
const int *pci;
int const *cpi;
const int const *cpci;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

These are: pointer to **int**, pointer to a **const int**, **const** pointer to an **int**, and **const** pointer to a **const int**, respectively. The same applies to **volatile** or combinations of **const** and **volatile** as in

```
const volatile int const volatile *cvpcvi;
```

Here, **cvpcvi** is a pointer to a **const volatile int**; that is, the object being pointed to cannot be used as an lvalue and it may be changed/used in some unknown way by an external process. The pointer itself is also **const** and **volatile**, meaning that the pointer cannot be used as an lvalue and may be used in unknown ways. All the possible permutations are valid although it is expected that only a few of them are useful. It is hard to see, for example, why you would want a **const** pointer to a **const** object, but who knows? So rather than arbitrarily restrict the valid combinations, ANSI C permits the programmer to do whatever makes sense for his or her particular environment, and that certainly is in keeping with "the spirit of C."

The Bottom Line

Since the Intel 8088 family of processors is commonly used for hosting free-standing systems written in C and in supporting interrupts, the capability of **const** and **volatile** is most welcome. Watch for their coming to a compiler in your area soon. (In fact, Wizard V3 already has it, as do others. Microsoft V3 reserves the keywords **const** and **volatile** for future use.)

Future Meetings

The schedule for future ANSI meetings is: September 14-18, Framingham, Massachusetts. The December 7-11 meeting scheduled for Phoenix, Arizona, has been replaced by the January 25-29, 1988, meeting at Austin, Texas. □

Rex Jaeschke is an independent computer consultant, writer, and lecturer. He is also a member of the X3J11 C Standard's Committee, Co-founder and Editor of the C Journal, *and the C Language Editor of the* DEC Professional.

If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to

Rex Jaeschke
2051 Swans Neck Way
Reston, VA 22091

Rex will address your concerns in future columns as space permits.

FREE SOURCE CODE!

VITAMIN C
It's good for your system!

FREE SOURCE CODE!

Vitamin C Difference

With **Vitamin C,** your applications come alive with windows that explode into view! Data entry windows and menus become a snap. Vitamin C's **open ended design** is full of "hooks" so you can "plug in" special handlers to customize most routines. Of course, Vitamin C **includes all source code FREE,** with no hidden charges. *It always has.*

Windows

Create windows with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, scroll bars, sizes to 32k, and more. Unique built-in feature lets users move and resize windows at run-time!

Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields, picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited validation via standard and user definable routines.

High Level Functions

Standard help handler provides context sensitive pop-up help messages any time the program awaits key strokes. So easy to use that a single function initializes and services requests by opening a window, locating, formatting, displaying and paging through the message.

Multi-level Macintosh & Lotus style menus make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens quickly and easily.

Text editor windows can be opened for pop-up note pads and general purpose editing. Features include insert, delete, word wrap, justify, cut, paste, search, and more!

VCScreen

With VCScreen and Vitamin C working together, you'll reach a new level or productivity you can't reach with a function library alone!

VCScreen speeds development even more! The interactive screen editor actually lets you draw input, output and constant fields, headings, boxes, lines, even a window for your froms to run in.

VCScreen generates readable C source code ready to "plug in" to your application and link with Vitamin C.

Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C $225.00
Includes ready to use libraries, tutorial, reference manual, demo, sample and example programs, and quick reference card; for IBM PC and compatibles. Specify compiler and version when ordering.

Vitamin C Source . . . . . FREE*
**Free with purchase of Vitamin C.*

VCScreen $99.95
Requires Vitamin C and IBM PC/XT/AT or true compatible.

Shipping $3 ground, $6 2-day air, $20 overnight, $30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on U.S. bank. Texas residents add 7¼% sales tax.

(214) 416-6447

creative PROGRAMMING
Creative Programming Consultants, Inc.
Box 112097 Carrollton, Texas 75011

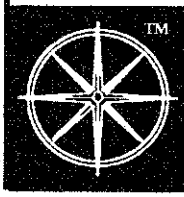
Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C Programmer's Toolbox

Productivity Enhancement Tools

CFlow	Generate a map of your program's subroutine hierarchy. Supports an unlimited number of program source listings.
CPrint	Beautify your program listings, making them consistant and easy to read.
PMon	Monitor your program execution and system calls, determining where the most time sensitive/critical sections are. Does not require any modification to an existing program.
*****	And more.... All from one source....
$79.95	30 day money back guarantee CA residents add applicable sales tax



MMC AD Systems
Box 360845
Milpitas, California 95035
(408)-263-0781

Number **94** on the Reader Service Card

DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 184 PAGE USER'S GUIDE

LIB + SRC $49

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 86 PAGE USER'S GUIDE

LIB + SRC $29

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $50 -- SRC $150

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $49 -- SRC $99

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT C (3.00 & LATER)

- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
P.O. BOX 21106
COLUMBUS, OH 43221
(614) 267 - 2143

Number **118** on the Reader Service Card

C CODE FOR THE PC

source code, of course

Panache C Program Generator	$150
QC88 C Compiler	$90
EMACS-like Editor	$75
TELE Kernel & Windows	$60
Make	$50
PC/MPX Multitasking Executive	$45
Coder's Prolog	$45
Biggerstaff's System Tools	$40
Translate Rules to C	$30
ICON String Processing Language	$25
Bison (YACC clone) & PREP	$25
LEX	$25
XLT Text Translator	$20

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders MasterCard/VISA

Number **30** on the Reader Service Card

TURBO SOURCE SEARCH

"BBS-by-mail"
2 FREE DISKS then just $2.50 A DISK

TURBO SOURCE SEARCH (TSS) is the largest SIG library of Turbo Pascal public domain/shareware programs.

TSS is in fact a Bulletin Board System (BBS). We have the usual modem-connect BBS (617-545-9131), but also offer a BBS-by-mail service where members may get the files from the library on disks by mail for just $2.50.

TSS members pay a one time fee of $25 and receive a set of catalog disks plus credit for 2 FREE DISKS.

TSS has over 110 disks, 35Mb of compressed files, of software. These files have been placed into public access channels by their authors and collected by TSS so that you can avoid the costs of collecting them yourself.

And since TSS is turning ONE YEAR OLD we are offering a one price special of $250 for full membership PLUS the full library of disks updated through disk 150 which should be added to the library by mid-summer!!! (Current members: $2 per disk, 50+ disk orders.)

For a set of catalog disks send $5, or send $25 to become a full member of TSS, the "BBS-by-mail" to:

TURBO SOURCE SEARCH
P.O. BOX 876, SCITUATE, MA 02066, 617-545-6677
data/BBS: 617-545-9131 (anyone may call!)

VISA, MASTERCARD, COD $5 extra, non-usa orders add $10

Number **35** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Character Tables and String Functions

by
William J. Hunt

The C library provides several useful functions for scanning character strings: strspn, strcspn, and strpbrk, for example. These functions are fine for casual use, but they may not be satisfactory where a number of strings must be scanned and quick execution matters. The pseudo-code below shows two possible implementations of strspn and the problems involved in each approach:

```
/* strspn(s1,s2) - finds the 1st char in s1 present in s2. */

/* First Implementation - two loops nested */
for each char in s1
  for each char in s2
    if( matched )
      return position in s1

/* Second Implementation - build table, then one loop */
char ct[256] ;
                              /* build a char table */
clear ct to zero              /* =0 default
for each char in s2  (s2[i])
  ct[ s2[i] ] = 1             /* =1 for chars in s2 */

for each char in s1 (s1[i])   /* now scan s1 */
  if( ct[ s1[i] ] != 0 )
    return( i )
```

The first implementation becomes inefficient when s2 contains more than a few characters; its execution time is proportional to the strlen(s1)* strlen(s2). The second implementation scans the s1 string efficiently, but unless s1 is quite long, setting up the char table (ct) is costly. We can improve on the second implementation by separating it into two functions: one to initialize a table and another to scan a string using an existing char table. This approach is very attractive when a table can be set up once and used to scan a string many times. The ctab.h, ctab1.c, and ctab2.c source files provide some useful functions based on char tables.

strspn, strcspn, and strpbrk are fine for casual use, but they may not be satisfactory where a number of strings must be scanned and quick execution matters.

The ctb__set function in the ctab1.c source file initializes a char table for scanning. For each char in a string, the corresponding element of the table is set to the value specified. If no string is specified, all char table elements are set to the value. Since char values may be signed (depending on the compiler, version, and option chosen), the logical and operation with the constant CHAR__MASK is needed to produce positive integer values between 0 and 255. Making a series of calls to ctb__set creates a char table with different values for different classes of characters; for example, the calls below set up a table to identify white space characters in a C source file:

```
ctb_set(ct,NULL,0) ;          /* non- white space chars */
ctb_set(ct," \t\n\f\v",1) ;   /* white space chars */
```

The ctb__spn function in ctab2.c scans a string using a char table already created. It returns the position of the first char whose table value equals a specified value. It requires a single array reference per char in the string whatever the table's contents. Using the char table defined above, the following example skips past white space in a string, finding the beginning and end of a non-white space series of characters:

```
i = ctb_spn(ct,s,0) ;  /* finds 1st non- white space char */
j = ctb_spn(ct,& s[i],1);/* finds next white space char */
```

A char table can also be used to translate C char strings efficiently. The ctb__xset function in ctab1.c sets up a char table for translation. For each char in the from string, the table element gives the character to which it is translated. For characters not in the from string, the table element is the same character (ct['a' & 0xff] = 'a' .)

The ctb__xlat function in ctab2.c translates a string in place using an existing char table. Like ctb__spn, it makes one table reference per character to be translated. The example below translates a string to lower case:

```
ctb_xset(ct,"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
                "abcdefghijklmnopqrstuvwxyz");
ctb_xlat(s,ct) ;
```

The char table functions presented are one application of the table lookup idea. Instead of executing an algorithm to decide what to do with each character, we build a table once and then just look the answer up in the table for each character to be processed. Table lookup algorithms are the fastest solution to many problems in data communications, numerical analysis, and compiler writing. □

William J. Hunt writes the Lattice C Users' Group *newsletter, where this material first appeared, and is also the author of* The C Toolbox, *published by Addison-Wesley. Readers may contact Bill at the following address:*

William J. Hunt
Lattice C Users' Group
P.O. Box 271965
Concord, CA 94527

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Code from Hunt, Character Tables...

```
/* ctab.h - defines and fun. prototypes for char table funs. */

/* for Lattice or MSC Ver. 1.xx or 2.xx, remove what's between */
/* the () */
int ctb_set(char *,char *,int) ;
int ctb_xset(char *,char *,char *) ;

int ctb_spn(char *,char *,int) ; /* like strspn() */
int ctb_xlat(char *,char *) ;    /* translates a char string */
```

```
/* ctab1.c - char table functions */
#include "stdio.h"

#define CTB_SIZE  256       /* size of char set (0-255) */
#define CHAR_MASK 0xff      /* char -> unsigned char mask */

int ctb_set(ct,s,val)       /* initialize char table entries */
 char ct[] ;                /* char table - one byte per char in set */
 char *s ;                  /* C char string */
 int val ;                  /* table value for chars in s */
 {
   register int i ;

   if( s == NULL )          /* if no string passed */
      { for(i=0; i< CTB_SIZE ; i++)
          { ct[i] = val ; } /*     initialize all entries */
      }
   else                     /* string passed */
      { while( *s != '\0' )/* for each char in the string */
          { i = *s & CHAR_MASK ; /* turn char into 0-255 value */
            ct[i] = val ;   /* set table = in for this char */
            s++ ;
          }
      }
 }

int ctb_xset(ct,from,to)    /* initialize char table for translation */
 char ct[] ;                /* char table - one byte per char */
 char *from ;               /* C string with chars to be translated */
 char *to ;                 /* C string - new values for corresponding */
                            /* chars in from string */
 {
   register int i ;

   for(i=0; i< CTB_SIZE ; i++)
      { ct[i] = i ; }       /* default value = no change */

                            /* now store from->to translation */
   while( (*from != '\0') && (*to != '\0') )
      { i = *from & CHAR_MASK ; /* turn char into 0-255 value */
        ct[i] = *to ;       /* set table = in for this char */
        from++ ; to++ ;
      }
 }
```

```
/* ctab2.c - Char table functions */
#include "stdio.h"

#define CTB_SIZE     256   /* size of char set */
#define CHAR_MASK    0xff  /* char -> unsigned char mask */

int ctb_spn(s,ct,val)       /* find 1st char with table entry = val */
 char *s ;                  /* string to search */
 char ct[] ;                /* char table */
 int val ;                  /* value to match */
 {
   register int i ;

   i=0 ;
   while( *s != '\0' )      /* look at each char in string */
      { if( ct[ *s & CHAR_MASK ] == val ) /* does table entry match ? */
          return( i ) ;  /*    matches val - return position */
        i++ ; s++ ;
      }
   return( -1 ) ;
 }

int ctb_xlat(s,ct)          /* translate a char string */
 char *s ;                  /* C string to be tranlated */
 char ct[] ;                /* char translation table */
 {
   while( *s != '\0' )
      { *s = ct[ *s & CHAR_MASK ] ;
        s++ ;
      }
 }
```


#1 C interpreter

C-terp

The professional C development environment

Your C compiler creates great final code . . . but as a programming tool, it's too, t o o s l o w. With C-terp you can edit, debug, and run without the wait. Nothing, but nothing, is faster for developing professional C programs.

Choose the perfect C-terp companion for your C compiler

C-terp/Microsoft	C-terp/XENIX
C-terp/Lattice	C-terp/Aztec
C-terp/Mark Williams	C-terp/C86

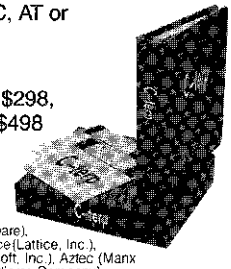
Link in all your compiler's functions, your own functions, add-on libraries, assembly routines, and data objects. Get instant access to everything in the C-terp interactive environment.

Only C-terp offers all this and more

- **Full K&R with common ANSI enhancements**
- **Source level interactive debugging**
- **Software paging for your big jobs**
- **Complete multi-module support**
- **Run-time pointer checking**
- **Unsurpassed reconfigurable screen editor**
- **Dual display and full graphics support**
- **Large model** ■ **Call-in**

ORDER C-terp TODAY (specify compiler)

C-terp runs on IBM PC, AT or compatibles.



Price:
MS-DOS 2.x and up - $298,
Xenix System V 286 - $498
MC, VISA, COD
30-day money-back
GUARANTEE

Trademarks: C-terp (Gimpel Software), C86 (Computer Innovations), Lattice(Lattice, Inc.), Xenix, Microsoft, MS-DOS (Microsoft, Inc.), Aztec (Manx Software), Mark Williams (Mark Williams Company), IBM (International Business Machines, Inc.)

GIMPEL SOFTWARE

3207 Hogarth Lane, Collegeville, PA 19426

(215) 584-4261

Contact this Advertiser directly

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MWWIND—A Menu with Windows Utility

by
Daniel A. Bronstein

Helping beginning and infrequent computer users determine which programs they should be using in different situations is easy when only floppies are involved. When a user is faced with a hard disk, however, things can become much more complicated. To help solve this problem when I turn my graduate students loose on my machine (18 megabytes of programs and 3 megabytes of data on a 22-megabyte hard disk), I wrote the MWWIND program. MWWIND will display up to a full screen of menu choices and supply up to 4 "pop-up" help windows in any location desired. The menu choices are implemented as batch files, each having a short name and each returning to the original directory and redisplaying the menu when done. This proved so effective that several of my colleagues have also installed MWWIND on their machines.

The basic outline of MWWIND is simple: It reads a file of text by redirection of input, displays the first part of the file (the main screen), locates the offsets in memory of each of the help windows, determines the letter which will call up each help window, and then sits and waits for keyboard input. Keyboard input of anything except <Esc> or '/' is passed through the keyboard buffer to DOS so that it can be executed. '/', when followed by the letter which calls up one of the help windows, displays that window. All four of the help windows can be displayed simultaneously. <Esc> "peels off" the last window put on the screen by redisplaying everything else.

The text file that is used has a very specific form. Optionally, it can start with '@' followed by a byte representing the row on which the cursor is to be placed after the main screen is displayed. If these two bytes are not found, MWWIND defaults to row 21. Immediately following this (or at the beginning if no '@' is found) must be single bytes representing the left hand column of the screen to be used (1 to 81, with 81 ['Q'] representing 0) and the top row of the screen to be used (1 to 24, or, again, 81 for 0). The text of the main screen immediately follows these two bytes. The default attribute (color) of the screen is 7 (normal monochrome), but this can be changed at any time by inserting an <Esc> byte (27) in the text of the screen, followed by a byte of the attribute (color) to be used until changed again. If there are help windows, the main screen text ends with a <cr><lf>; if there are no windows, it ends with a byte of 244 (an arbitrary choice).

Each window starts with a byte of 244, a lower case [a-z] letter to be used for calling up that window, a byte (1 to 81 with 81 representing 0) giving the left hand column of the window and a byte (2 to 24) giving the top line of the window. Rows 0 and 1 are reserved for MWWIND's own help message. These bytes are followed by the text of the window in the same format as for the main screen, with desired attribute (color) changes. Each window except the last must end with a <cr><lf>; the last one ends with a byte of 244.

The code up to VSM1 simply determines the video board in use and stores the appropriate addresses for later use. The screen is then cleared and the text file read into STORAGE at the end. If a cursor exit row is specified, it is stored, and then the text of the main screen is transferred to the video board starting at SHOW1. The actual transfer is done by the MOVWIND procedure, which itself calls the MOVLINE procedure. The code from WINDOW to OUT locates and stores the offsets of the windows and the letters that will call them up. The code from OUT to the end of the main program waits for keyboard input and then either transfers the buffer contents to DOS or displays or peels off a window.

The MOVWIND procedure first resets ES to the appropriate video board segment and DI to 0 and then calculates the addition to DI needed to make the screen or window appear at the proper screen location (through MW1). It then calls MOVLINE, which reads the line from STORAGE (MOVEIT through MOVE4), and, after checking the video retraces (MOVE5, MOVE6), moves one byte at a time to the video board address via the STOSW in MOVE6. MOVEWIND then calculates the location of the start of the next line and either moves DI there if it is showing a window or fills the intervening addresses with spaces if it is showing the main screen. The latter is necessary to overwrite any window that might have been in that area of the screen. Before returning, MOVWIND resets ES to the current segment so that DOS will behave properly when the user exits the program.

The BACKUP and CALCWIND procedures determine which windows are currently active and the order in which they were activated so that they can be peeled off if the user wants to do that.

The first version of MWWIND used BIOS services for the screen writing, but that turned out to be too slow as the window capacity increased, so the program was rewritten in its present form to write directly to the video memory. In order to make the program run well on older color/graphics boards, I included code for testing the 6845 video controller chip; if you do not anticipate facing such a problem, you can eliminate the lines in the listing whose comments end with a pound sign (#). As currently presented, MWWIND does not run in EGA modes; the addition of the needed 5 or 6 lines of code is "left as an exercise for the reader." Similarly, it should be relatively simple to add the code for additional windows if you want them. □

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Daniel A. Bronstein is a professor of law who has given up crossword puzzles for the intellectual challenge of programming in assembly and Turbo Pascal. He can be reached at

Department of Resource Development
Michigan State University
East Lansing, MI 48824-1222

Code from Bronstein, MWWIND... 1

```
;                       MWWIND.ASM
;    Displays a full screen of text and up to four windows,
;  all contained in a text file which is read by redirection of input.
;  Copyright Daniel A. Bronstein, 1986, all commercial rights reserved.

;*********************************************************************
; Requires that the text file to be read be in the                   *
; following form ([] is optional):  ['@' followed by                 *
; a byte of 0-24] - cursor row on exit, default is 21.               *
; Main screen of text and attributes.  Must start with               *
; a byte representing the left hand column in which the              *
; screen starts (1-81, 81 = column 0) and a byte                     *
; representing the top row of the screen (1-24, or 81 =              *
; row 0).  Default attribute is 7; can be changed at any             *
; place by an <Esc> byte (27) followed by a byte of the              *
; new attribute.  Main screen ends with <cr><lf> unless             *
; there are no windows, in which case it ends with a byte            *
; of 244.                                                            *
; [Windows] up to 4 of them.  Each window MUST start on a            *
; new line with a byte of 244, a lower case (a-z) letter             *
; which calls it up, a byte (1-81) telling the program the           *
; left hand column of the window (81 = column 0), and a              *
; byte (2-24) telling the program the top row of the window          *
; (rows 0 & 1 are reserved for the program). The actual text         *
; of the window follows the same rules as the main screen.           *
; All windows end with <cr><lf>, except the last one, which          *
; ends with a byte of 244.                                           *
;                                                                    *
; In operation, the main screen displays first.  Windows are         *
; called by typing a '/'followed by the letter that calls up         *
; that window.  Windows are "peeled off" by typing <Esc>.            *
; Multiple windows can be displayed simultaneously.  Typing          *
; anything except '/' or <Esc> exits and passes the typed            *
; character on to DOS.  Typing '/' followed by an invalid            *
; window calling letter exits without passing anything to DOS.      *
;*********************************************************************
  ;*************************************************************
  ; The lines with # at the ends of the comments can be omitted *
  ; if you do not have problems with "snow" when writing to the *
  ; video board in your machine. They do slow the program down. *
  ;*************************************************************

cseg    segment para    public  'CODE'
        assume  cs:cseg,ds:cseg
        org     100h

start:  jmp     begin


wmsg    db      9,0,27,112,'<Esc> to remove last window; '
        db      'Progname <Enter> to run a program;',13
        db      '      /[a-z] for help windows; '
        db      ' any other key to exit.          ',244,'$'
 wmsgend:
         monscr  dw      0B000h          ;Address of mono screen
         m6845   dw      03BAh           ;Address of 6845 chip for mono        #
         cgscr   dw      0B800h          ;Address of cg screen
         cg6845  dw      03DAh           ;Address of 6845 chip for cg          #
         mainflg db      0               ;Flag for main screen show
         one60   dw      160             ;Constant used later
         two     dw      2               ;Constant used later
         chread  dw      0               ;Number of bytes in file
         wind1   dw      0               ;Address (offset) of window1 text
         wind2   dw      0               ;   "       "     "  window2  "
         wind3   dw      0               ;   "       "     "  window3  "
         wind4   dw      0               ;   "       "     "  window4  "
         w1key   db      0               ;Calling key for window1
         w2key   db      0               ;   "     "   "  window2
         w3key   db      0               ;   "     "   "  window3
         w4key   db      0               ;   "     "   "  window4
         windctr db      0               ;Number of windows in text file
         swcol   dw      0               ;Starting column * 2 storage
         colcnt  dw      0               ;Column counter on display
         outrow  db      21              ;Cursor row on exit storage
         vidseg  dw      0               ;Video board address storage
         add6845 dw      0               ;6845 address storage                 #
         copyrt  db      'Copyright Bronstein, 1986$'
         modemsg db      13,10,'Works only in 80x25 TEXT MODES.',13,10,'$'
         filemsg db      13,10,'Cannot find/open/read named file.',13,10,'$'
         wdctr   db      0               ;Windows Displayed counter
         dispw1  db      0               ;Number of first window displayed
         dispw2  db      0               ;   "   "  second    "       "
         dispw3  db      0               ;   "   "  third     "       "
         dispw4  db      0               ;   "   "  last      "       "

 begin:  mov     ah,15                   ;Get current display mode -
         int     10h                     ;returns AL=mode.
```

TRUE
MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - Intertask message communication. Send/receive/check message present on 64 message queues.
  - Task control by means of semaphores. Get/release/check semaphores.
  - Change priority-256 priority levels.
  - Suspend task for specified interval.
  - Spawn and terminate external and internal tasks.
  - Disable/enable multitasking.
  - and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: $24.95 OR $99.95 with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number 109 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        cmp     al,7                    ;If al=7, is mono,
        je      vsmono                  ;so jmp to calculate its location -
        cmp     al,2                    ;if al= 0 or 1, not 80x25,
        jb      vs05                    ;so is error,
        cmp     al,3                    ;if al=2 or 3, is cg 80x25,
        jbe     vs1                     ;so jmp to calculate location,
vs05:   jmp     vserr                   ;otherwise is an error.
vs1:    mov     ax,cgscr                ;Cannot address board directly, so
        mov     bx,cg6845               ;move the cg screen and 6845          #
        jmp     vsm1                    ;addresss to ax and bx
vsmono: mov     ax,monscr               ;or mono addresss to ax
        mov     bx,m6845                ;and bx, then                         #
vsm1:   mov     vidseg,ax               ;store the result.
        mov     add6845,bx              ;                                     #
        mov     ax,0600h                ;Clear the screen using the
        mov     bh,07                   ;BIOS int 10h function 6 (bh=7 for
        mov     cx,0000                 ;normal attribute, cx=0 upper left of
        mov     dx,2480h                ;screen, dx=2480 lower right).
        int     10h
        cld                             ;Clear the dir flag for loading the
        mov     cx,sp                   ;text and load text at end (storage)
        sub     cx,100h                 ;using DOS function 3Fh
        sub     cx,dx                   ;with bx=0; i.e., DOS redirection of
        mov     dx,offset storage       ;input.
        mov     bx,0
        mov     ah,3Fh
        int     21h
        jnc     go_on                   ;If no carry flag, success, so go on,
        jmp     fileerr                 ;else a file error.
go_on:  mov     chread,ax               ;Store number of bytes read.

;*************************************SHOW - Display main screen.
show:   mov     si,offset storage       ;Set up to read from storage the number
        mov     cx,chread               ;of bytes read in by redirection.
        mov     ah,7                    ;Assume normal attribute (07)
        lodsb                           ;and load first byte.
        cmp     al,'@'                  ;If is '@', next byte is the row
        je      getrow                  ;for the cursor on exit, so jmp.
        dec     si                      ;Otherwise decrement si to reset to
        jmp     show1                   ;start and jmp to show it.
getrow: lodsb                           ;GETROW - Read exit cursor row
        dec     cx                      ;(remembering to decrement the counter)
        mov     outrow,al               ;and store it.
show1:  mov     mainflg,1               ;SHOW1 - Set flag to tell movwind this
        call    movwind                 ;is main screen and call the procedure.
        mov     mainflg,0               ;Reset main flag to 0.
        cmp     wdctr,0                 ;If no windows showing, jmp to window,
        je      window                  ;else use backup procedure to show the
        call    backup                  ;windows other than the last one,
        jmp     out                     ;then jump to get kbd input.

;*************************************WINDOW - Now get the window addresses
window:                                 ;and keys.
        cmp     windctr,0               ;If windctr > 0 we've already loaded
        je      w05                     ;the window keys and addresses, so
        jmp     out                     ;jmp to out.
w05:    inc     windctr                 ;Increment the window counter,
        cmp     windctr,1               ;if >1, jmp to w2,
        ja      w2                      ;else
        lodsb                           ;load byte
        dec     cx                      ;and decrement counter.
        cmp     al,26                   ;If is end of file
        je      out                     ;jmp to get kbd input.
        mov     w1key,al                ;The byte in al is the key to call
                                        ;this window and
        mov     wind1,si                ;the address in si is its location.

w1:     lodsb                           ;W1 - Load next byte.
        cmp     al,244                  ;If is 244 we're at start of
        je      w05                     ;next window so jmp,
        cmp     al,26                   ;if end of file, we're done so jmp,
        je      out                     ;else
        loopnz  w1                      ;loop back for next byte.
w2:     cmp     windctr,2               ;W2 - If window counter > 2 jmp
        ja      w3                      ;to w3, else
        lodsb                           ;load next byte
        dec     cx                      ;and decrement counter,
        mov     w2key,al                ;store the calling key and
        mov     wind2,si                ;the starting address and
```



```
        jmp     w1                      ;jmp back to reload.
w3:     cmp     windctr,3               ;W3 - Works same as w2.
        ja      w4
        lodsb
        dec     cx
        mov     w3key,al
        mov     wind3,si
        jmp     w1
w4:     cmp     windctr,4               ;W4 - works same as w2.
        ja      out
        lodsb
        dec     cx
        mov     w4key,al
        mov     wind4,si
        jmp     w1

;*************************************OUT - Read kbd for next action.
out:    mov     dl,0                    ;First get cursor out of the way
        mov     dh,outrow               ;to column (dl) 0, row (dh) specified
        mov     ah,2                    ;at begining (or default of 21) via
        int     10h                     ;function 2 of int 10h.
        mov     ah,1                    ;Check kbd buffer for input (function
        int     16h                     ;1 of int 16h) and, if none, go back
        jz      out                     ;and check it again. If a key is there,
        cmp     al,'/'                  ;is it '/' or Esc (27)? If yes, jmp to
        je      out1                    ;out1 to clear the buffer, else jmp to
        cmp     al,27                   ;outout to pass the buffer contents to
        je      out1                    ;the system for other action after exit.
        jmp     outout

out1:   mov     ah,0                    ;OUT1 - Clear kbd buffer (function 0
        int     16h                     ;of int 16h).
        cmp     al,'/'                  ;If is '/',
        je      out1                    ;go back to out1 for next key.
        cmp     al,27                   ;If is not Esc (27) jmp to out15,
        jne     out15                   ;else redisplay the main screen by
        jmp     show                    ;going back to let show do the work.
out15:  cmp     al,90                   ;OUT15 - Is the char lower case?
        ja      out2                    ;If yes, jmp to out2
        add     al,32                   ;else make lower case by adding 32.
out2:   cmp     al,w1key                ;Is it the key for window 1? If not
        jne     out3                    ;jmp to check for next window; if yes,
        mov     si,wind1                ;move address of window to si,
        call    movwind                 ;call move window procedure,
        mov     dl,1                    ;mov the window number (1) to dl
        jmp     out9                    ;and jmp to out9.
out3:   cmp     al,w2key                ;Same thing for window 2.
        jne     out4
        mov     si,wind2
        call    movwind
        mov     dl,2
        jmp     out9
out4:   cmp     al,w3key                ;Same thing for window 3.
        jne     out5
        mov     si,wind3
        call    movwind
        mov     dl,3
        jmp     out9
out5:   cmp     al,w4key                ;Check for window 4 key. If not there,
        jne     outout                  ;exit by jmp to outout, else show
        mov     si,wind4                ;window 4.
        call    movwind
        mov     dl,4
out9:   cmp     wdctr,0                 ;OUT9 - If no windows yet shown, move
        ja      out91                   ;the window number in dl to
        mov     dispw1,dl               ;dispw1 for storage, then jmp to out99,
        jmp     out99                   ;else check to see if is 2nd window.
out91:  cmp     wdctr,1                 ;Same as out9 but for 2nd window shown.
        ja      out92
        mov     dispw2,dl
        jmp     out99
out92:  cmp     wdctr,2                 ;Same as out9 but for 3rd window shown.
        ja      out93
        mov     dispw3,dl
        jmp     out99
out93:  mov     dispw4,dl               ;Store 4th window number.
out99:  xor     dl,dl                   ;OUT99 - Clear dl,
        mov     si,offset wmsg          ;show the message window at the
        call    movwind                 ;top of screen using move window,
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        inc     wdctr                   ;increment the windows displayed counter
        jmp     out                     ;and jmp back to out.

outout: mov     dl,0                    ;OUTOUT - Set exit column to 0 and
        mov     dh,outrow               ;row to [outrow] or default of 21
        mov     ah,2                    ;via function 2 of
        int     10h                     ;int 10h,
exit:   int     20h                     ;then exit to DOS.

vserr:  mov     dx,offset modemsg       ;VSERR - Display the video mode error
        mov     ah,9                    ;message via function 9
        int     21h                     ;of DOS and then
        jmp     exit                    ;exit.

fileerr:mov     dx,offset filemsg       ;FILEERR - Display the file error
        mov     ah,9                    ;message via function 9
        int     21h                     ;of DOS and then
        jmp     exit                    ;exit.


;*******************************PROCEDURES************************************

                                        ;MOVLINE PROCEDURE - reads and moves
movline         proc    near            ;one line to the video board.
        push    dx                      ;Save dx and then move the              #
        mov     dx,add6845              ;address of the 6845 there.             #
moveit: lodsb                           ;MOVEIT - Load byte.
        cmp     al,244                  ;If is 244 is start of next window
        je      m_out                   ;so jmp out.
        cmp     al,27                   ;If is Esc next byte is new attribute
        jne     move1                   ;value, else jmp to move1.
        dec     cx                      ;Decrement counter and
        lodsb                           ;load byte, then
        mov     ah,al                   ;transfer it to ah (attribute location),
        dec     cx                      ;decrement counter,
        jmp     moveit                  ;and go back for next byte.

move1:  cmp     al,13                   ;MOVE1 - If cr, is end of line,
        je      m_out                   ;so jump out.

move3:  cmp     al,10                   ;MOVE3 - If not lf,
        jne     move4                   ;jmp to move4, else
        dec     cx                      ;decrement counter and
        jmp     moveit                  ;ignore the lf and get next byte.

move4:  inc     colcnt                  ;MOVE4 - Increment the column counter,
        push    ax                      ;save the char and attribute and        #
move5:  in      al,dx                   ;test the retrace of the screen.        #
        test    al,1                    ;                                       #
        jnz     move5                   ;If not 1, jmp back to wait, else       #
        cli                             ;clear the interrupt,                   #
move6:  in      al,dx                   ;test for the other retrace             #
        test    al,1                    ;                                       #
        jz      move6                   ;waiting if not ready.                  #
        pop     ax                      ;Restore the char and attribute and     #
        stosw                           ;move them to the monitor board.
        sti                             ;Restore the interrupt                  #
        dec     cx                      ;and loop back
        jmp     moveit                  ;for next byte (decrementing counter).
m_out:  pop     dx                      ;-- M_OUT -- Pop dx and                 #
        ret                             ;return, passing the contents
movline         endp                    ;of al (either 244 or 13).

;*************************************MOVWIND PROCEDURE - Moves whole window
movwind         proc    near            ;to video board memory.
        push    ax                      ;Save the registers
        push    bx
        mov     es,vidseg               ;Set es:di to video board memory
        mov     di,0                    ;address for moving.
        lodsb                           ;Load first byte (column number),
        cmp     al,81                   ;if 81 (Q), col = 0,
        jne     mw05                    ;so
        mov     al,0                    ;replace the 81 with 0.
mw05:   cbw                             ;Expand al to ax,
        mov     bx,ax                   ;and move it to bx,
        mul     two                     ;multiply by 2 to account for attribute
        mov     swcol,ax                ;bytes and store result in swcol.
        xor     ax,ax                   ;Clear ax
mw07:   lodsb                           ;and load next byte.
```



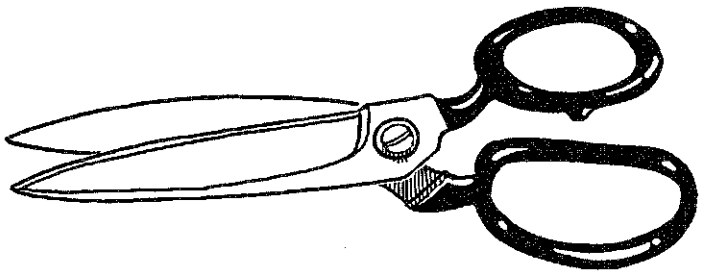
```
        cmp     al,81                   ;Again, if 81 (Q),
        je      mw1                     ;is 0, so jmp over next few steps.
        cbw                             ;Expand al to ax and
        mul     one60                   ;multiply by 160 (character + attribute)
        add     di,ax                   ;bytes in a line, add the result to di
        xor     ax,ax                   ;to get starting location and clear ax.

                                        ;MW1 --
mw1:    mov     colcnt,bx               ;Move start column (bx) to the column
        add     di,swcol                ;counter, add swcol to di to get full
        call    movline                 ;start location, and call MOVLINE.  If
        cmp     al,244                  ;return in al is 244, end of window so
        je      mw_out                  ;jmp out.  Only other return is al=13,
        cmp     colcnt,80               ;so check the column count. If < 80, jmp
        jb      mw2                     ;to mw2 to pad it, else
        jmp     mw1                     ;go back for next line.

mw2:    cmp     mainflg,1               ;MW2 - If this is main screen jmp to
        je      mw3                     ;do the space padding.
        push    ax                      ;Pad out line.  First save ax
        mov     ax,80                   ;(since ah has current attribute) then
        sub     ax,colcnt               ;subtract colcnt from 80,
        mul     two                     ;multiply by 2 (byte + attribute),
        add     di,ax                   ;move there by adding result to di,
        pop     ax                      ;and restore ax,
        jmp     mw1                     ;then jmp back for next line.

mw3:    push    cx                      ;MW3 - Pad with spaces (32) so as to
        mov     cx,80                   ;cover the deleted windows.  Save cx,
        sub     cx,colcnt               ;subtract colcnt from 80,
        mov     al,32                   ;move space char (32) to al
        rep     stosw                   ;repeat the move of the space and the
        pop     cx                      ;current attribute, retrieve cx,
        jmp     mw1                     ;and jmp back for next row.

mw_out: assume  es:cseg                 ;MW_OUT -- Restore es to current
        pop     bx                      ;segment, restore the registers,
        pop     ax
        ret                             ;and return to caller.
movwind         endp

;*************************************BACKUP PROCEDURE - Peel off last window.
backup  proc    near
        cmp     wdctr,4                 ;If 4 windows active, jmp to bu1
        je      bu1
        cmp     wdctr,3                 ;If 3 windows active, jmp to bu2
        je      bu2
        cmp     wdctr,2                 ;If 2 windows active, jmp to bu3
        je      bu3
        cmp     wdctr,1                 ;If 1 window active, jmp to bu_out
        je      bu_out
        jmp     buout1                  ;If 0 windows active, jmp to return.
bu1:    mov     dispw4,0                ;BU1 - Erase number of last window,
        mov     dl,dispw3               ;mov the number of wind to be shown to
        call    calcwind                ;dl for passing to calcwind.
bu2:    mov     dl,dispw2               ;BU2 - works same as BU1.
        call    calcwind
bu3:    mov     dl,dispw1               ;BU3 - works same as BU1.
        call    calcwind
bu_out: dec     wdctr                   ;BU_OUT - decrement the window counter
buout1: ret                             ;and return to caller.
backup  endp

;*************************************CALCWIND PROCEDURE - Redisplay a window.
calcwind        proc    near
        cmp     dl,1                    ;If passed number in dl > 1, jmp, else
        ja      cw2                     ;mov address (offset) of wind1 to si and
        mov     si,wind1                ;then jmp to cw9 to show it.
        jmp     cw9
cw2:    cmp     dl,2                    ;Same, except for dl > 2.
        ja      cw3
        mov     si,wind2
        jmp     cw9
cw3:    cmp     dl,3                    ;Same, except for dl > 3.
        ja      cw4
        mov     si,wind3
        jmp     cw9
cw4:    mov     si,wind4                ;If dl = 4, mov wind4 to si to show it.
```


1987 Programmer's Journal 5.4

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Bronstein code, Continued 6

```
cw9:    call    movwind                 ;CW9 - Call movwind and
        xor     dl,dl                   ;zero out dl, then
        mov     si,offset wmsg          ;show the message window at the
        call    movwind                 ;top of screen.
        ret
calcwind        endp

;*****************************END OF PROCEDURES*********************************

storage label   byte                    ;Location for storing the text file.

cseg    ends
        end     start
```



A TEXT EDITOR MORE PEOPLE SHOULD USE.

dBASE Programmers

are you ready?

dB→c

"C" IS READY FOR YOU!

dB2c Offers:

- Complete implementation of dBASE III+ functions and commands in C source code.
- Interfaces for Lattice's dBC and Faircom's c-tree.
- Complete source code
- Support for the following C Compilers: Microsoft, Lattice, Aztec, DeSmet, and Instant-C.
- The freedom and peace of mind that only source code can offer.
- Works on color, monochrome, and EGA monitors.

dB2c Toolkit $299.

DAVID J. MARSH

Call or Write:
SOFTWARE CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

Number **93** on the Reader Service Card

dBASE to C

the dBx™ Translator

- dBx produces quality C direct from dBASE II, III, or III+ programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- Includes full screen handler and allows a choice of C database managers.
- May be used to move programs, help dBASE programmers learn C easily, or as a daily tool.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA or any standard C system.
- Priced from $350; also available from distributors.

from **Desktop Ai** 1720 Post Road East, Westport, CT 06880
Telephone: **203-255-3400** • Telex: **6502972226MCI**
MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate dBx is a trademark of Desktop Ai

Number **103** on the Reader Service Card

PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

Periscope III - debugger with 64K protected RAM and breakout switch; breakpoints for hardware, memory, port, data. Real-time trace buffer, pass counter. PC $ 829

AI-Expert System Dev't

Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
SQL Dev't Package	MS	$ 229
Auto-Intelligence	PC	$ 739
Experteach - Powerful, samples	PC	$ 339
Exsys	PC	$ 309
Runtime System	PC	$ 469
Insight 2+	MS	$ 379
Intelligence/Compiler	PC	$ 739
T.I.: PC Easy	PC	$ 435
Personal Consultant Plus	PC	$2589
Personal Consultant Runtime	PC	$ 85
Turbo Expert-Startup(400 rules)	PC	$ 129
Corporate (4000 rules)	PC	$ 359

AI-Lisp

Microsoft MuLisp 85	MS	$ 159
PC Scheme LISP - by TI	PC	$ 85
Star Sapphire	MS	$ 459
TransLISP - learn fast	MS	$ 89
TransLISP PLUS		
Optional Unlimited Runtime		$ 139
PLUS for MSDOS		$ 179

Others: IQ LISP ($239), IQC LISP ($269)

AI-Prolog

APT - Active Prolog Tutor - build applications interactively	PC	$ 49
ARITY Prolog - full, 4 Meg		
Interpreter - debug, C, ASM	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
Standard Prolog	MS	$ 77
MacProlog Complete	MAC	$ 269
MicroProlog - Prof. Entry Level	MS	$ 85
MicroProlog Prof. Comp./Interp.	MS	$ 439
MPROLOG P550	PC	$ 175
Prolog-86 - Learn Fast	MS	$ 89
Prolog-86 Plus - Develop	MS	$ 229
TURBO PROLOG by Borland	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS_PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 89
Basic Windows by Syscom	PC	$ 95
BetterBASIC	PC	$ 129
Exim Toolkit - full	PC	$ 39
Finally - by Komputerwerks	PC	$ 85
Inside Track	PC	$ 50
Mach 2 by MicroHelp	PC	$ 55
Peeks n Pokes	PC	$ 35
QuickBASIC	PC	$ 69
Stay-Res	PC	$ 75
Turbo BASIC - by Borland	PC	$ 69

FEATURE

NET-TOOLS - Access NETBIOS-compatible network systems from *Microsoft* C, Pascal, FORTRAN, Assembler, Lattice C. Full Source. No Royalties PC $ 129

FREE Newsletter

Insightful commentary, guest columnists, survey results, and valuable resource listings. Interviews, technical articles, predictions — even cartoons. No wonder 96% of our readers pass *The Programmer's Letter* on to their friends; no wonder 72% make sure they get their copy back to keep for reference! You can request a FREE sample copy today by calling our toll-free number. A personal subscription is just $25 per year.

Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7 PM to 7 AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
C86 PLUS - by CI	MS	$379
Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev'rs Kit	PC	$ 77
Datalight Optimum - C	MS	$109
with Light Tools by Blaise	PC	$168
Lattice C - from Lattice	MS	$269
Let's C Combo Pack	PC	$ 99
Let's C	PC	$ 59
Microsoft C 4.0- Codeview	MS	$275
Rex - C/86 by Systems & Software - standalone ROM	MS	$695
Turbo C by Borland	PC	$ 69
Uniware 68000/10/20 Cross Compiler	MS	Call
Wizard C	MS	$299
Rom Development Package	MS	$259

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$219
C Trainer - by Catalytix	PC	$ 89
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs	MS	$369
Interactive C by IMPACC Assoc.	PC	$209
Run/C Professional	MS	$155
Run/C Lite	MS	$ 79

C Libraries-General

Blackstar C Function Library	PC	$ 79
C Essentials - 200 functions	PC	$ 75
C Function Library	MS	$109
C Tools Plus (1 & 2) - Blaise	PC	$125
C Utilities by Essential	PC	$129
C Worthy Library - Complete, machine independent	MS	$249
Entelekon C Function Library	PC	$119
Entelekon Superfonts for C	PC	$ 45
Greenleaf Functions-portable, ASM		$139
LIGHT TOOLS by Blaise	PC	$ 69

Atari ST & Amiga

We carry full lines of Manx, Lattice, & Metacomco.

Call for a catalog, literature and *solid* value

800-421-8006

THE PROGRAMMER'S SHOP

Your complete source for software, services and answers

5-J Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 5/87

RECENT DISCOVERY

CxPERT - Expert systems shell, translates to C code to integrate with your application. Certainty factors, explanations, inheritance, frames, help MS $269

dBASE Language

Clipper compiler	PC	Call
dBASE II	MS	$329
dBase III Plus	PC	$429
dBASE III LanPack	PC	$649
DBXL Interpreter by Word Tech	PC	$139
FoxBASE+ - single user	MS	$349
Quick Silver by Word Tech	PC	$439

dBASE Support

dBase Tools for C	PC	$ 65
dBrief with Brief	PC	Call
DBC ISAM by Lattice	MS	Call
dBx Translator to C	MS	$319
dFlow - flowchart, xref	MS	Call
Documentor - dFlow superset	MS	Call
Genifer by Bytel-code generator	MS	$299
QuickCode III Plus	MS	$239

Fortran & Supporting

50:More FORTRAN	PC	$ 99
ACS Time Series	MS	$399
Forlib+ by Alpha	MS	$ 59
MS Fortran - 4.0, full '77	MS	$279
No Limit - Fortran Scientific	PC	$115
PC-Fortran Tools - xref, pprint	PC	$165
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	$139

MultiLanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N-multiuser	MS	$455
Flash-Up Windows	PC	$ 79
GSS Graphics Dev't Toolkit	PC	$375
HALO Graphics	PC	$205
Development Package	MS	$389
Informix 4GL-application builder	PC	$789
Informix SQL - ANSI standard	PC	$639
Opt Tech Sort - sort, merge	MS	$119
PANEL	MS	$215
Pfinish - by Phoenix	MS	$229
PolyLibrarian by Polytron	MS	$ 79
PolyBoost - speed I/O, keyboard	PC	$ 69
PVCS Corporate-source control	MS	$309
PVCS Personal	MS	$109
QMake by Quilt Co.	MS	$ 79
Report Option - For Xtrieve	MS	$109
Screen Machine	PC	$ 59
Screen Sculptor	PC	$ 95
SRMS - source control	MS	$107
Synergy-create user interfaces	MS	$375
VXM - multi-env. link	MS	$175
Xtrieve - organize database	MS	$199
ZAP Communications - VT 100	PC	$ 89

FEATURE

UI Programmer - Quickly generate dBASE User Interfaces, prototypes. Use supplied templates or create own. Pop-up help, bounce bar menus, screen forms. II, III, FoxBASE+, Quicksilver, Clipper. PC $249

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item.

Number **114** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Variable Passing and Hiding in QuickBASIC 2 Subprograms

by
Bernard H. Robinson, Jr.

One criticism that BASIC bashers love to level at the language concerns the global range of BASIC's variables. The support of subprograms with private variables in Microsoft's QuickBASIC 2 compiler goes a long way towards eliminating this criticism. There are four ways of passing variables from the calling program to subprograms and vice versa, all of which need to be well understood for efficient use of the compiler. In this article, I have tried to concentrate on material that I found confusing when I started using QuickBASIC 2, and I have treated lightly things that are well explained in the manual. Even those of you who already have some experience using the compiler may find this material helpful because it collects into a few pages information that is scattered throughout the compiler manual, sometimes in ambiguous language.

In the following, I use "BASICA" as shorthand for Microsoft BASIC interpreters for the IBM PC and compatibles in general, neglecting the differences among them, and "QB2" for Microsoft QuickBASIC 2 (I'm using version 2.01). Just as I wrote this article, I learned that Version 3 of QuickBASIC, with some major improvements over Version 2.01, will be available about the end of this month (April 1987). I don't expect that the information in this article will be significantly different for Version 3.

Subroutines versus Subprograms

In addition to supporting the familiar subroutines of BASICA, QB2 also supports "subprograms." Although QB2 subprograms are almost always used as subroutines, we will follow the practice of the QB2 manual and use the term "subprogram" to distinguish such subroutines from ordinary subroutines. Let's take a minute to contrast the two.

Subroutines

In BASICA, subroutines share all variables with the main program. A subroutine is properly entered with a GOSUB statement and exited with a RETURN statement. A subroutine is defined by ending a block of statements with a RETURN statement. If the program flow comes to a subroutine, the subroutine is executed whether or not a GOSUB was executed.

BASICA-type subroutines operate in QB2 with the following differences from the way they do in BASICA. QB2 supports alphanumeric line labels as well as line numbers, so you will probably want to give the beginning line of each subroutine a descriptive alphanumeric label instead of a line number. An alphanumeric line label can be up to 40 characters long, any mixture of letters and numbers, and must end with a colon. Experiment shows that you can use periods in a line label, but I can't find any documentation for this.

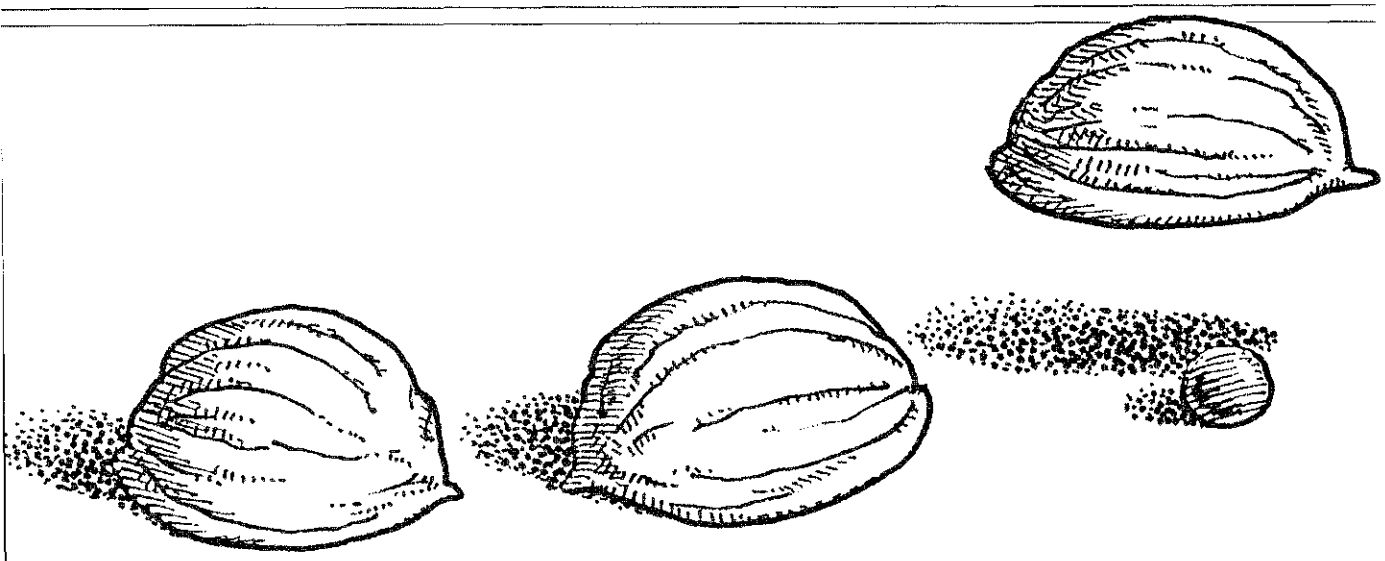
A subroutine in the main part of a program shares variables with the main program and other subroutines in the main program. A subroutine inside a subprogram shares only those variables visible to the subprogram. A subroutine in a subprogram can be entered only from within that subprogram. Subroutines in the main part of the program cannot be entered from a subprogram.

Subprograms

All variables in the main program are hidden from those in a subprogram and vice versa, except the variables the programmer made visible. Variables in one subprogram are hidden from variables in other subprograms. Subprograms must be entered with CALL statements and exited with END SUB or EXIT SUB statements. Subprograms can be called from the main part of the program or from other subprograms, either directly or through some other subprogram which calls it, but a subprogram cannot call itself. Subprograms are defined by subprogram "declarations," which must start with a SUB statement followed by a block of statements defining the function of the subprogram and end with an END SUB statement.

Since a subprogram is executed only by a CALL statement, the program execution flow skips over subprogram declarations. A subprogram declaration does not have to come before the call to the subprogram, so the programmer has considerable freedom as to where in the program subprograms are declared; however, a subprogram cannot be declared inside another subprogram. The placement of subprogram declarations in a program should be such that the program is easy to read and understand. It would be illogical to place subprogram declarations in the middle of a FOR NEXT loop, inside a multiline IF THEN block, etc. Often, grouping subprogram declarations near the beginning of the program file is a good choice. The subprogram declarations can be in a separate file that is included into the compilation with a "$INCLUDE" metacommand in the main program. Subprogram declarations can even

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



be in a separate program module that is compiled separately and put in a library. An advantage of QB2 is the relative ease with which subprograms can be assembled into libraries so that those developed for one program can be reused in later programs.

In addition to the subprograms discussed so far, QB2 can also use subroutines written in assembly language and assembled. Assembly language subroutines are beyond the scope of this article.

Like BASICA, QB2 supports one-line functions defined by the DEF FN statement. New in QB2 is the support of multi-line, user-defined functions. In contrast to subprogram variables, all main program variables are shared with the user-defined functions except variables that are declared STATIC (private) in the function definitions. The QB2 manual cautions the user about special circumstances that may result in unexpected side effects with user-defined functions in QB2. My personal preference is to use subprograms instead of user-defined functions unless there is a very good reason to want the function format, but this may be sheer idiosyncrasy on my part. In any case, user-defined functions are not treated further here.

Preliminaries

Before we discuss subprograms in more detail, it will help to define some of the terms that will be used and present a little background material.

Variables

The term "variable" will be used in its general sense—any numeric or string variable, including arrays and individual array elements. QB2 variable names can be up to 40 characters long. Letters (upper- or lowercase), numbers, and periods can be in the name, but the name must start with a letter. Variable names are not case-sensitive—ABC, abc, and Abc would all be considered the same variable by the program.

Program Modules and the $INCLUDE Metacommand

The term "program module" is used here to mean all the source code that is compiled to form a single .OBJ module. A QB2 program module may consist of more than one disk file, thanks to the $INCLUDE metacommand. A $INCLUDE metacommand causes the referenced file to be read into the module at the $INCLUDE metacommand during compilation just as though the code had been written in the module originally.

Metacommands are instructions to the compiler to perform some action during compilation. Metacommands are well described in section 10.3 of the manual. The following is a sample $INCLUDE metacommand:

```
REM $INCLUDE: 'FISUB.BAS'
```

where "FISUB.BAS" is the name of the program file to be included in the compilation.

To clarify a bit further, let's consider a simple example. Suppose we are working on a program named FINANCE.BAS and we decide it is convenient to place a number of subprograms in a separate file called FISUB.BAS. We would then put a $INCLUDE metacommand in FINANCE.BAS to include FISUB.BAS. The effect of the $INCLUDE metacommand is to make FISUB.BAS part of FINANCE.BAS as far as the QB2 compiler is concerned. It is appropriate to say that FINANCE.BAS and FISUB.BAS are in the same program module even though they are in separate disk files. When FINANCE.BAS is compiled, the code in FISUB.BAS will be part of the resulting .OBJ file. Calls in FINANCE.BAS to subprograms in FISUB.BAS are definitely within the same compiled module.

Later, once the subprograms are well debugged, we might prefer to compile FISUB.BAS separately and place the resulting .OBJ file in a user library file as described in Chapter 7 of the QB2 manual. The subprograms in FISUB.BAS will no longer be in the same compiled module with FINANCE.BAS. Depending upon the methods we used to pass variables between FINANCE.BAS and FISUB.BAS, this change in status may require some small changes to one or both program files. As a minimum, we will have to remove the $INCLUDE metacommand that included FISUB.BAS in FINANCE.BAS.

Sometimes we use the term "external module" to mean a separately compiled program module.

Main Program and Main Module

We need a way to refer to the main part of a QB2 program that is not inside subprogram declarations or user-defined function definitions. I am going to use "main program" for this.

In the FINANCE program example above, we eventually had

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

a program module named FINANCE.BAS and another module, FISUB.BAS, that was compiled separately and put in a library. In this case, we could call FINANCE.BAS the main module of the program. FINANCE.BAS and FISUB.BAS would each have a "main program" section by our definition.

The Calling Program

A call to a subprogram can be made from the main program or from within another subprogram. Sometimes when we speak of "the calling program," we will be referring to the main program even if a call is made from a subprogram. If we speak of "the COMMON statements in the calling program," we have to be referring to the COMMON statements in the main program; COMMON statements cannot appear inside a subprogram.

Static and Dynamic Arrays

QB2 supports two kinds of arrays, static arrays and dynamic arrays. It is only when you need to redimension arrays or deallocate arrays to recoup memory space during the running of a program that the difference becomes important. The fundamental difference between the two lies in the way memory space is assigned to them. Both static and dynamic arrays can be passed to subprograms and back with some minor differences in syntax, which are discussed below.

Arrays are declared as static or dynamic as follows: If an array is dimensioned with constant subscripts, it is a static array by default. If an array is dimensioned with variable subscripts, it is a dynamic array by default. These defaults can be overridden by $STATIC or $DYNAMIC metacommands. If the $STATIC metacommand is given, arrays dimensioned in following DIM statements are compiled as static arrays. If the $DYNAMIC metacommand is given, the arrays in following DIM statements are compiled as dynamic arrays. Like BASICA, QB2 allows an array with a maximum element number (subscript) of 10 to be used without a DIM statement; this is called "implicit dimensioning." In QB2, all implicitly dimensioned arrays are compiled as static even if a $DYNAMIC metacommand is in effect.

Static Arrays: Space is assigned for static arrays at compile time, and static arrays cannot be redimensioned during the running of the program. If the ERASE statement is executed for a static array, all elements are set to zero if it is a numeric array or to null if it is a string array. If a static array is to be listed in a COMMON statement, the DIM statement for the array must be placed before the COMMON statement. Static arrays occupy slightly less space and can be accessed faster than dynamic arrays.

Dynamic Arrays: Space is allocated for dynamic arrays at run time. Dynamic arrays make more efficient use of memory space in that they are allocated only when needed and can be redimensioned with a REDIM statement or deallocated with an ERASE statement. If an ERASE statement has been executed for a dynamic array, a DIM statement must be executed to redimension the array before it can be used again. A DIM statement for a dynamic array is considered an executable statement and must appear after all COMMON statements in a program. If a dynamic array is named in a COMMON statement, its name

FORTRAN PROGRAMMERS

Looking for the right PC FORTRAN language system? If you're serious about your FORTRAN programming then you should be using F77L - LAHEY FORTRAN.

"Lahey's F77L FORTRAN is the compiler of choice. It's definitely a 'Programmers FORTRAN,' with features to aid both the casual and the professional programmer. . . F77L compiled the five files in a total of 12 minutes, which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN." - PC Magazine

Compare the features and performance of other PC FORTRANs with F77L and you will find that F77L is clearly the superior product.

- Full Fortran 77 Standard (F77L is not a subset)
- Popular Extensions for easy porting of mini and mainframe applications
- COMPLEX*16, LOGICAL*1 and INTEGER*2
- Recursion - allocates local variables on the stack
- IEEE - Standard Floating Point
- Long variable names - 31 characters
- IMPLICIT NONE
- Fast Compile - Increases productivity
- Source On Line Debugger (Advanced features without recompiling)
- Arrays and Commons greater than 64K
- Clear and Precise English Diagnostics
- Compatibility with Popular 3rd Party Software (i.e. Lattice C)
- Easy to use manual
- Technical Support from LCS

- *NEW FEATURE - NAMELIST*

F77L - THE PROGRAMMER'S FORTRAN

$477.00 U.S.

System Requirements: MS-DOS or PC-DOS, 256K, math coprocessor (8087/80287)

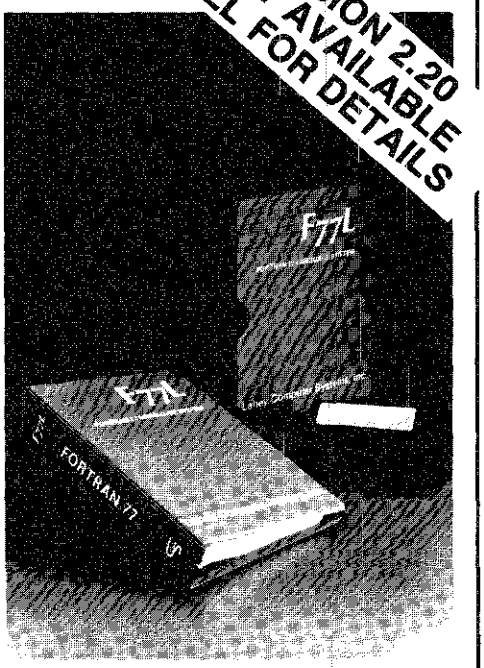
FOR MORE INFORMATION: (702) 831-2500

Lahey Computer Systems, Inc.
P.O.Box 6091
Incline Village, NV 89450
U.S.A.

International Dealers:
England: Grey Matter Ltd., Tel: (0364) 53499
Denmark: Ravenholm Computing, Tel: (02) 887249
Australia: Computer Transitions, Tel: (03) 537-2786
Japan: Microsoftware, Inc., Tel: (03) 813-8222

SERVING THE FORTRAN COMMUNITY SINCE 1967

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation.Pro FORTRAN refers to Professional FORTRAN a trademark of International Business Machines.



Editor's Choice
- PC Magazine

Number 44 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

must be followed by "(N)," where N is a constant stating the number of dimensions of the array. If a DIM statement is executed for a dynamic array that is already allocated, a "Redimensioned array" error message is given.

Methods of Variable Passing

These four methods of variable passing are available in QB2 and will be discussed below:

1. Passing in the argument list of SUB and CALL statements.
2. Passing with the SHARED attribute in the calling program.
3. Passing with the SHARED statement in the called subprogram.
4. Passing with COMMON statements in both the calling program module and an external module defining subprograms.

These methods are often used in combination, with some variables being passed one way and some another.

Variable Passing with the Argument List

The most straightforward way to pass variables to subprograms is to include them in the argument list of the CALL and SUB statements. This method follows the same rules whether the called routine is in the same compiled module or in a separately compiled module in the user library. This is the best method to use if the same subprogram is called repeatedly with different sets of variables.

Declaring a Subprogram

A subprogram declaration must start with a SUB statement and end with an END SUB statement. The format of the SUB statement is as follows:

```
SUB name[ (list) ] STATIC
```

The components of the SUB statement are:

SUB keyword identifying the statement.

name name of the subprogram, up to 31 characters long. I can't find any documentation for which characters are allowable in a name. All examples I've noted in the manual use alphanumeric characters with a letter first. My experiments seem to indicate that periods are allowed in the name.

list optional list of arguments (variables) passed to the subprogram. If present, the list is enclosed in a set of parentheses and the arguments are separated by commas.

STATIC keyword required in the format.

The format of a CALL statement is as follows:

```
CALL name[ (list) ]
```

The components of a CALL statement are:

MULTITASKING AND MULTIUSER PERFORMANCE FOR 27¢ PER DAY

With **OPERATING SYSTEM TOOLBOX,** the Software Construction set that allows you to build your own custom operating system.

With **PCNX,** Wendin's PC version of Bell Laboratories' incredible UNIX Operating System.

With **PCVMS,** A system similar to the VAX/VMS Mainframe Operating System that brings an elegant set of services to the PC.

FROM WENDIN . . . of course!

EACH PRODUCT IS PRICED AT ONLY $99.00, AND INCLUDES SOURCE CODE ON DISK.
If you average that cost out, that's only 27¢ per day for one year.
THE REST OF YOUR LIFE IT'S FREE!

★★★ **MENTION WHERE YOU SAW THIS AD** ★★★
WHEN YOU ORDER AND RECEIVE FREE A
HIGH QUALITY PEN AND PENCIL SET IN A DELUXE GIFT BOX
COMPLIMENTS OF WENDIN.

ORDER HOTLINE (509) 624-8088

Wendin is a registered trademark of Wendin, Inc. PNX, PCVMS, and Operating System Toolbox. are trademarks of Wendin, Inc.

WENDIN®
BOX 3888
SPOKANE, WA 99220-3888

Working beyond the horizon to develop the operating systems of tomorrow
© Copyright 1987 Wendin, Inc. (509) 624-8088

Number 77 on the Reader Service Card

WINDOWS – MENUS – DATA ENTRY – SCREENS

HI-SCREEN XL™

Application Developers! **HI-SCREEN** *XL*™, the newest version of HIGH SCREEN, is a complete and unmatched programming tool for managing your user-interface. **HI-SCREEN** *XL*™ features a revamped and enhanced screen editor, increased speed and performance, along with a new thorough reference and user's manual.

Whatever tool you own: compare, you'll be convinced!

3. **DATA ENTRY**
Just define your data entry fields and let HIGH SCREEN™ do the job: -integer/real/alphabetical/date/chosen characters. format/range/upper case-lower case convertion/justify left/right/center/size/color. required input/auto-tabbing/password/Help message. Automatic error handling. Cursor and arrow keys management. Field by field or full screen checking mode for increased flexibility.

1. **WINDOWS:** for on-line help, menus, data entry. Up to 26 levels deep.
2. **MENUS:** pop-up, pull-down or Lotus-style, also for batch files.
3. **DATA ENTRY:** automatic field checking: format, type, range... field by field or full screen. Error handling.
4. **SCREENS:** full-featured screen editor. Language independent. Fast display from RAM.

Royalty free, not copy protected, 30 day money back guarantee, trade up available.

Software, Inc. (415) 397-4666
PC/SOFT Product Line
500 Sutter St. Suite 222
San Francisco, CA 94102
HI-SCREEN *XL*™ is $149
(CA res. add tax), S&H USA $5
Visa, M/C welcome

PASCAL, C, dBASE, BASIC, COBOL, FORTRAN, ASSEMBLER, etc.

Number 110 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

CALL keyword identifying the statement.

name name of the called subprogram (see above).

list list of arguments, if any. The number of arguments present must be the same as the number in the SUB statement. Each argument must be the same type (integer, single precision, double precision, or string variable) as the corresponding argument in the SUB statement.

Here is an example of a subprogram declaration:

```
SUB FUN.ONE( X, Y!, N%, Z#, W(2), V#, S$  ) STATIC
(lines of code defining the action of the subprogram)
END SUB
```

Here is an example of a call to the above subprogram:

```
CALL FUN.ONE( A, B!, C%, D#, F(), G#(6), S1$ )
```

The above illustrates variable passing by means of the argument list of the SUB and CALL statements. Note that the variable names in the CALL statement do not have to be the same as those in the SUB statement though they could be if desired. The order of the variables in the argument list determines which variable in the calling program represents the corresponding variable of the subprogram. This makes it easy to call a subprogram using several sets of variables in the calling program. Note that each of the variables in the sample CALL statement must agree in type with the corresponding variable in the subprogram declaration. If desired, DEFtype statements can be used in the calling program to declare the type of variables in the CALL statement, but note the differences between the way the compiler and the interpreter process DEFtype statements as described in the QB2 manual.

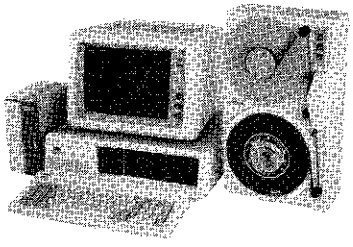
Caution: While DEFtype statements can be placed in a subprogram declaration to declare the type of variables within the subprogram, these DEFtype statements will not apply to the arguments of its SUB statement; arguments in the SUB statement must always be typed with type characters (%, !, #, and $). If no type character is placed on a variable in a SUB statement, the compiler will take it as a single precison numerical variable by default.

In "W(2)" in the SUB statement, the "2" indicates that "W( )" is a two-dimensional array; note that in the CALL statement, the parentheses of the corresponding argument are empty. This is the method of sharing the entire array with the subprogram. In contrast, the argument "G#(6)" of the CALL statement shares a single element, "G#(6)," of the double precision array with the subprogram. In the corresponding position in the SUB statement, only a double precision variable, "V#," is used to represent this single element. Note that a single element of an array is not allowed in a SUB statement.

Passing by Reference

The default for variable passing in QB2 is two-way; the calling program variables in the argument list are passed to the subprogram, and changes to corresponding variables in the sub-

9-Track Tape Subsystem
for the IBM PC/XT/AT
XENIX or
MS-DOS.



The solution to your micro/mainframe communications problem is available today!

Qualstar's new ½ inch 9-track MINISTREAMER™ brings full ANSI data interchange capability to the PC. Now you can exchange data files with virtually any other computer using 9-track tape.

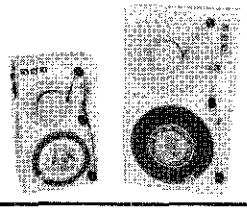
Available in both 7" and 10½" versions, the MINISTREAMER weighs in at only 27 pounds and uses less desk space than an ordinary sheet of paper, yet provides full 1600/3200 BPI capability at an affordable price. Up to 134 megabytes of data (depending on format) can be stored on a standard 10½" reel of tape, thus making the MINISTREAMER a highly-reliable answer to your backup requirements as well.

Tape subsystem includes tape drive, coupler card, cables, dust-cover and MS-DOS or XENIX compatible software. Prices start at $2,995.

386 READY!

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!



QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822

Number **97** on the Reader Service Card

$99 **BUSINESS BBS**

24 hour business information center via modem (300-2400 baud). Setup custom multi-level menus, data entry forms and info-sheets easily with no programming req'd. Integrated data base, XMODEM up/downloads, remote PC operation. With source code $249.

$99 PC **COMMX** $119 CP/M

Emulates: VT100/102, Wyse, HP, ADM, TV, IBM, ADDS
Transfers: KERMIT, XMODEM, COMMX mainframe, TLX/TWX. POPUP hotkey to DOS or programs. Unattended control scripts, dial directory for 700 entries & electronic mail features.

$59 **C DATA ENCRYPTION**

Data Encryption Standard (U.S. government standard FIPS PUB46) in Microsoft "C". Includes compression & telecomm formatting, allowing faster transmission & less storage space plus compatibility with any computer or service. Complete "C" source code $249.

HAWKEYE GRAFIX Inc
BOX 1400, OLDSMAR FLORIDA 33557
DIAL 813-855-5846

Number **57** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

program change the variables in the calling program. This is called "passing by reference" because it is accomplished internally by sending the subprogram the address of the variables in the calling program.

Passing by Value

Sometimes we would like to pass the subprogram the current values of certain variables in the calling program without allowing the subprogram the capability to change the value of the variables in the main program. This is called "passing by value." The only way to do this in QB2 is to enclose the variable in the calling program argument list with an extra set of parentheses; for example, suppose the program fragment below:

```
A = 5 : B = 4
CALL DOIT( (A), (B), C )
PRINT "C = ";C
```

calls the following subprogram:

```
SUB DOIT( X, Y, Z)
STATIC Z = X + Y
END SUB
```

The subprogram DOIT( ) has access to the values of A and B and can compute the value of Z. Since the values of A and B were passed by value, if changes are made to X and Y in the subprogram, values of A and B in the main program will not be affected. The value of Z can be passed back to the calling program as the value of C since the variable C was passed by reference rather than by value.

None of the other ways of passing variables described below allows passing by value, but see below on the passing of expressions.

Passing the Value of Expressions

It is also possible to pass the value of expressions such as (X + Y) or (Y * Z) in the argument list of CALL statements by enclosing them in parentheses. Expressions are always passed by value. Only a single variable of the proper type is put in the corresponding position in the SUB statement to receive the value of the expression. None of the other ways of passing variables discussed below allows passing expressions. Note that expressions are not allowed in SUB statements.

Passing Arrays in Argument Lists

To pass an entire array to a subprogram, put the name of the array in the CALL statement argument list with "( )" appended to the name to indicate that it is the name of an array. In the corresponding position in the SUB statement, an array name must appear with "(N)" appended, where N is a constant stating the number of dimensions of the array. Don't confuse the number of dimensions of the array with the maximum element number of the array!

When an array is passed in the argument list, don't dimension the array in the subprogram. The subroutine can find the upper and lower bounds of an array by using the LBOUND and UBOUND functions, or you can pass the bounds to the subprogram.

Continued on Page 57

QuickBASIC just got quicker with
QuickPak

QuickPak is a superb collection of enhancements, subroutines, and instructional material designed to help you get the most out of programming in BASIC.

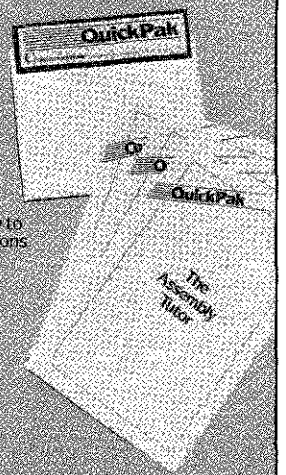
- Powerful assembly language routines to give your programs more speed, more power, and full access to DOS and BIOS services. SORT all or part of a string array with one command! Complete windowing capability — display help screens instantly, overlay text. FIND any string or sub-string within an entire array *regardless of capitalization* — accepts wildcards. READ directories into your programs from any drive or path. READ/WRITE disk sectors — create your own DOS utilities! MANY, many more programs included.
- Professionally written QuickBASIC routines and functions. Powerful input routines for text, dates, and numbers. Menus, scroll bars, date/time functions, and much more.
- The Assembly Tutor — a complete guide to learning assembly language *from a BASIC perspective*. Learn how to create your own routines and extensions.
- Tips and Tricks book — packed with clever ideas and techniques to help *you* be a better programmer.

You get all this, all of the source code for *every program included*, and a thirty-day money back guarantee for only $69.00.

No royalties are required for using any of the QuickPak routines in your programs. Not copy protected, of course.

by
CRESCENT SOFTWARE
64 Fort Point Street, East Norwalk, CT 06855
(203) 846-2500

QuickPak requires Microsoft QuickBASIC or BASCOM, DOS 2.0 or higher. Visa, MC, C.O.D., or checks accepted.



QBase
Quick screens and more for BASIC

If you program in BASIC, then QBase is your shortest route to professional titles, data entry screens, and even complete applications. Unlike ordinary screen builders, QBase is fast, powerful, and very easy to operate. Pull-down menus and function keys are used throughout, and context sensitive help is always available. (Though you probably won't ever need it!) QBase lets you draw boxes, copy and paste blocks, color any portion of the screen, and design custom help windows. Creating input fields is as simple as answering a few questions. If all you need are quick screens and windows, then look no further.

Make complete applications

But QBase is much more than just a screen builder, because it can also create an entire application. Data entry, editing, searching, and relational file operations are all handled automatically. Any number of databases may be combined in a single program, with more than fifteen screens per database.

QBase can do all this because it's really two separate programs. The screen builder lets you create and edit screens, which are saved as *form definitions* that any program can readily access. A run-time package (easily modified) then does all the work of managing the database.

Everything is included

All of the source code for the screen editor and run-time program is included, thoroughly documented to show how they work, and where they may be customized. QBase also includes The Hardware Tutor — a clear and practical discussion of electronic concepts and circuits, but written from a programmer's perspective. QBase is available for either Microsoft QuickBASIC or Borland Turbo Basic (please specify when ordering). It costs only $99.00 and comes with a money-back satisfaction guarantee. No royalties, not copy protected, of course.

CRESCENT SOFTWARE
64 Fort Point Street
East Norwalk, CT 06855 (203) 846-2500

QBase requires DOS 2.0 or higher, and at least 256K. (512K recommended.)
Visa, MC, COD, checks accepted.

Number 83 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

DIRECT MAIL ORDER

ORIGINAL DESIGN T-SHIRTS

BORN TO PROGRAM PJT101

NOW! Soft Wear from Programmer's Journal!

Only $9.95 each, Postage paid. Classic PJ cartoons & cover graphics. Reproduced as dynamic Multi-color imprints on high quality 50/50 cotton T-Shirts. In sizes S, M, L, XL. Born to program? Send for yours today!

RS-232 HEAD PJT103

PARALLEL PORTS PJT102

COMMON C BUG PJT104

BACK ISSUES

•Vol. 1, No. 1: Norton batch files and Pascal I/O; Program style manual; COBOL on the IBM PC; Benchmarks: System analysis; BASIC modules; Rounding error.

•Vol. 1, No. 2: Defining character sets; C86 BIOS calls; Troubleshooting; Copy protection: Pascal code; Input routines; Writing an assembler.

•Vol. 1, No. 4: Diskette directory via Pascal; MASM WHILEWEND; C/ASM interface; ASM stay resident routine; Structured programming; Software law; Pascal tutorial; Default values and variable names.

•Vol. 1, No. 5: Multi-tasking software review; Transporting C programs; Programmer's calculator; dBASE II bargraphs; Utilities; ROM BIOS equates; Pascal check program; Combining modules in Assembler; Structured programming in BASIC; Agreement outline.

•Vol. 2, No. 1: Freelance consulting; Sorting in C; Pascal text editor source code; CRT distress; Compiled BASIC; Assembly modules for C; Designing a relational database; Ultra Utilities: Punctuation and Style reviewed.

•Vol. 2, No. 2: Interview with Brian Kernighan; Understanding C through assembly language; Utility routines in C and assembly; Music in C; Automatic line number optimizing in BASIC; Assembly language tutorial: Full-screen input routines in BASIC; dBASE II printer control codes; Pascal tools reviewed; C programming support products.

•Vol. 2, No. 3: Software development environments; Creating a productive screen handling tool; Assembly tutorial; Generating graphics within windows; Composition and editing system for musical ideas; Hybridizing assembly and BASIC; Creating screens; Linking Assembler to Pascal and BASIC.

•Vol. 2, No. 4: Writing a custom ROM BIOS for PC DOS; RISC, Transputers, and mass storage; The future of disk protection; 8088 assembly macros; Random access files in BASIC; C utility routines you can buy; Using the PMATE macro language; Choosing a database manager; E/SP syntax-checking; Pascal editor reviewed.

•Vol. 2, No. 5: Developing the APX CORE Executive; From Assembler to C via a 211 byte .EXE main( ); Scroll control using BASIC and Assembler; Faster screen handling with BASIC and ASM; Random access files, Part 2; RTOOL, a PC-resident programmer's toolkit; dBkey reviewed; Pascal's pointer power.

•Vol. 2, No. 6: Periscope—building a hardware board for the IBM PC; Intel 80286 bugs by FNE; AT&T surprises all; Field Usage and pointer subtraction in C; Calendar functions for BASIC; Memory segmentation; Reduce your .EXE file sizes; Enhancing Debug; dBASE II to dBASE III? THINKTANK reviewed.

•Vol. 3, No. 1: November's Comdex; C—its uses and abuses; Using the C preprocessor; Memory segmentation examples; Index files, Part 1; Structured programming language; Periscope, a symbolic debugger.

•Vol. 3, No. 2: UNIX! (again); Artificial Intelligence; Doctor C's tips—poiniers to functions; Calendar date routines in BASIC; Calculate record layouts, automatically!

•Vol. 3, No. 3: NAPLPS and the professional programmer; Instant-C; Printer control from a batch file; Repositioning the stars in Starfinder; Reducing EXE file size through data positioning; tools for better programming.

•Vol 3, No. 4: More UNIX stuff; C programming books, Part 1; C code excerpts, Part 1; How to cure the "dreaded locked keyboard" condition; Calling MS DOS from Turbo Pascal; DOS 2.x sort and find filters; the SET Tools.

•Vol 3, No. 5: Windows done here; C programming books, Part 2; C code excerpts, Part 2; A menu-driven PC using DISPLAY.COM; Turbo Pascal and MASM: making them work together; dBASE II routines; Turbo Pascal 3.0; TopView Programmer's Toolkit.

•Vol. 3, No. 6: About C's HLL overhead; How to recover all your costs; Report on the Comdex and Usenix shows; The Santa Cruz Operation XENIX for the IBM PC; Common C bugs; CTOOLS2; Turbo Pascal 3.0 and subdirectories; More dBASE II routines; Clipper reviewed; A better way to interface BASIC and Assembler.

•Vol. 4, No. 1: FNE on PC/AT hard disk problems and plunging 64K DRAM prices; AUTOINC explained; Sorting using an AVL tree; Combining binary trees; Using DOS to call programs from Turbo Pascal; An operating system for the people; Final installment of Brett Salter's dTOOLS; Two microcomputer graphics books reviewed.

•Vol. 4, No. 3: FNE—another RISC machine, the paging game, software viewpoint, programmer talent, and Aloha, 32032; Sun Microsystems' next generation PC; Consulting—reusable modules; INT24.PAS, a critical error handler for Turbo Pascal; Memory model independence in Assembler; An approach to windowing for the assembly language programmer; Proposed ANSI C Standard, Part 2; Microsoft and Lattice C 3.0 reviewed; Accelerator boards for software development reviewed.

•Vol. 4, No. 4: Hal W. Hardenbergh on adult toys, tomorrow's micros, HALGOL update, benchmarks, and RISCs; Frank D. Greco on parallel computers, RISC, UNIX; Consulting—efficient custom programming; DOS 2 file handles and file I/O; Programming the WordStar document file conversion utility; More optimizing for speed; Teach an old compiler a new trick; Memory resident programs in C; Tips on formatting with BASIC PRINT USING statements; The structure of dBASE II and III file headers.

•Vol. 4, No. 5: Hal W. Hardenbergh on 32-bit PCs, the 68040, Weitek's data-parh part, etc.; Frank D. Greco on large screen monitors, language translators, UNIX, and more; Prolog intro, Turbo Prolog reviewed; Consulting—tax traps, consulting styles, nest eggs; ASM—Search and Display; ASM—the GROUP statement; The proposed ANSI C Standard, Part 3; C—measuring stack usage and Lattice C's startup code; Short boards reviewed.

•Vol. 4, No. 6: DTACK GROUNDED; Unix Books. Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate and Stay Resident; Turbo Pascal inline code; Pascal—Assembler souce code for DOS function calls; Alignment problems in C and managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

•Vol. 5, No. 1: DTACK GROUNDED; The new DOS; Marketing and rules to live by; File redirection; Inside the EGA; String allocation in Pascal; ANSI C Language Standard, Part 5; More tips on formatting BASIC data.

•Vol. 5, No. 2: DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

•Vol. 5, No. 3: Starting your own software house—publicity; Unix books, part 2; Consulting—merger mania; EGA data control; Modula-2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and Forcing File Updates; Formatting BASIC output with buffered PRINT USING.

To order, just use the convenient form located on the perf insert, at right. If the card is missing, you may place credit card orders by calling our office at:

(503) 484-2162

PRODUCED 2008 BY UMZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Variable Passing with the SHARED Attribute, Same Compiled Module

When several variables must be passed to a number of different subprograms declared in the same program module, passing all the variables in the argument list may become clumsy due to the length of the list. Then it may be convenient to use the SHARED attribute to pass some or all of the variables. The SHARED attribute must be used with one or more COMMON, DIM, or REDIM statements by placing the word SHARED following the keyword COMMON, DIM, or REDIM on statements in the main program. If not needed for their usual function, COMMON and DIM statements can be placed in a program just for the purpose of sharing variables. Using the SHARED attribute on a statement in the main program does not require a matching statement in the subprograms. All the variables named in the statement or statements having the SHARED attribute will be global to all of the subprograms in the same program module. In contrast to the argument list method of passing variables, in passing variables with the SHARED attribute, the variable names must be the same in the calling program and the called subprogram.

Although DIM statements are provided in BASIC to dimension arrays, among the sample programs supplied by Microsoft with QuickBASIC Version 2.01, both PPRINT.BAS and REMLINE.BAS contain DIM statements with ordinary variables (that is, not arrays) in them for the purpose of sharing them with subprograms. I have been unable to find any documentation for this practice in the QB2 manual, but it appears to function properly, at least in Version 2.01 of QB2. It would seem more appropriate to use COMMON statements for this purpose; a named COMMON block could be used if it were desirable to keep the COMMON used for this purpose separate from COMMON statements used for program chaining or variable passing to separately compiled subprograms.

See the discussion above of static and dynamic arrays for special considerations depending upon the type of array when naming arrays in COMMON statements.

Overriding the SHARED Attribute, Same Compiled Module

Since the SHARED attribute makes the named variables global to all the subprograms in the same compiled module, there is a possibility that the programmer may need to deny some of the subprograms access to some of the shared variables. The STATIC statement may be used in these subprograms. The STATIC statement consists of the word STATIC followed by a list of variables that are to be made private to the subprogram, overriding the SHARED attribute. In addition to being private to the subprogram, variables listed in a STATIC statement will retain the last value assigned to them so that the next time the subprogram is called, the variable will still have the value it had at the end of the previous call. The STATIC statement is sometimes used just to make a variable retain its last value even when it is not needed to make the variable private.

If you need to place an array name with the STATIC statement, the array name must be followed by "(N)," where N is a constant stating the number of dimensions of the array, and the STATIC statement must be followed immediately by a DIM statement dimensioning the array in the subroutine.

Note that the STATIC statement can be used only in subprogram declarations and in multi-line function definitions.

Copy, Delete, & Display your files by date

Ever wanted to copy **only** those files you changed today? Or do you need to delete **last** week's data files? Maybe you need to copy all files created last year to a backup disk. If so you need **THE DATE UTILITIES!** With these routines you can manipulate files created **on, before, or after** a specified date. All utilities support paths and full **wildcards** for file names.

☞ **DCOPY** copies files based their name and date. It offers optional prompt before copying and archive bit control. DCOPY pauses when a floppy becomes full and lets you insert a new floppy or format a floppy before continuing.

☞ **DERASE** lets you delete a file or group of files based on their name and date. A prompt before erase mode is also offered.

☞ **DDIR** displays a disk directory limited by file name and date. The double column directory can be sorted based on name, extension, date, or file size.

☞ **CDATE** changes the date and time stamp of a file. An optional prompt before change mode is also offered.

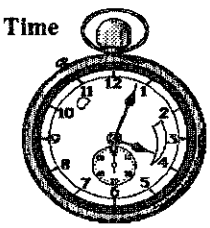
THE DATE UTILITIES package costs $47.50. No charge for shipping in the U.S. The package includes DCOPY, DERASE, DDIR, and CDATE routines along with bound documentation. These routines are **not** copy protected. IBM PC or compatible, DOS 2.1 & up.

VISA 312-377-3520 MC

ECI P.O. Box 61, Wasco, Illinois 60183-0061

Number 70 on the Reader Service Card

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

SRMS™
Software Revision Management System

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without conflict
- Windowing Shell interface to simplify use
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+, 224 K, F/H Disk

SRMS Version 3.0.....$185

QMAKE™
Intelligent Program Generation Utility

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX makefile compatability, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+, 128K F/H Disk

QMAKE Version 1.2.....$99

SRMS + QMAKE $250

NEW ! TXT Tools NEW !

QSE - Quilt Text Stream Editor
QSRCH - Quilt File Search Utility (Like UNIX GREP)
QDIFF - Quilt Windowing File Difference Utility

TXTTOOLS Version 1.0..... $85

QUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

Number 46 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Even though a variable is declared STATIC in a subprogram, the variable can still be placed in the argument list of a CALL statement in the subprogram and passed to another subprogram and can even be changed if the corresponding variable is changed in the called subprogram.

Caution: Do not declare a variable STATIC in a subprogram if the variable is in the argument list of the subprogram SUB statement. If you do, the STATIC declaration will block the action of the variable in the argument list, but the compiler will not give an error message to warn you of this.

Variable Passing with SHARED Statement, Same Compiled Module

We have just considered a method, used in the main program, which makes specified program variables global to all subprograms in the module. In contrast, the method about to be described is used in a subprogram declaration to make the main program share specified variables with it, without making the variables available to other subprograms. The SHARED statement (not to be confused with the SHARED attribute) can be used only inside subprogram declarations. The SHARED statement consists of the keyword SHARED followed by a comma-separated list of variables. Any array names in a SHARED statement must be followed by an empty set of parentheses "( )" to indicate an array. Both ordinary variables and array names can appear in the same SHARED statement.

As in the method using the SHARED attribute, the variable names must be the same in the calling program and the called subprogram.

It is worth emphasizing that the SHARED statement in a subprogram shares the listed variables in the subprogram with the main program only, even when the subprogram is called from another subprogram. Let me explain with an example. Suppose we have two subprograms in the same program module that are as follows:

```
SUB XYZ STATIC
Q = 5
CALL ABC
END SUB

SUB ABC STATIC
SHARED Q
PRINT "Q = ";Q
END SUB
```

Since XYZ calls ABC in the same module and since ABC contains "SHARED Q," it might appear that Q would be passed to ABC, but it won't be since the Q in ABC is shared only with the main program and the Q in XYZ is not shared with the main program. If a change is made to make the Q in XYZ available to the main program, such as in adding a "SHARED Q" statement in XYZ, the Q of XYZ will be available to ABC by way of the main program.

Of course, in a real program, a subprogram such as ABC that has only one argument would most likely have been declared with an argument list to pass the argument. In that case, XYZ could have called ABC by putting Q in the argument list without making Q available to the main program.

Variable Passing with Paired COMMON Statements, Separately Compiled Modules

We previously discussed the use of the SHARED attribute on COMMON, DIM, or REDIM statements to share variables in the main program with subprograms in the same compiled module. We will now consider a somewhat similar method, used only to call subprograms compiled in an external module. This method uses pairs of COMMON or named COMMON statements only. Aside from passing variables in the argument lists, this is the only way to share variables with subprograms compiled in external modules and put in a user library.

In this method, one or more COMMON statements are put in the calling program module naming the variables to be shared. Corresponding COMMON statements with the SHARED attribute on them are put in the program module containing the subprogram declaration. Although the SHARED attribute is required in the module declaring the subprograms, the calling program COMMON statements may or may not have the SHARED attribute; the SHARED attribute should be used in the calling program module only if it is needed to share variables with subprograms within the calling module. Named COMMON blocks may be used to pass different sets of variables to different separately-compiled subprogram modules.

Each external module defining subprograms would have only the COMMON and named COMMON statements needed to name the variables being shared by that module. Be careful with blank COMMON blocks. COMMON statements are called "blank COMMON statements" when they are not named COMMON statements. When more than one blank COMMON statement appears in a program module, the whole group of blank COMMON statements make up the blank COMMON block. The order of variable names in this whole block determines which variable in the calling program corresponds to a variable in the external module. Thus blank COMMON statements in the called module must be in the same order in which they appear in the calling program. The blank COMMON block in the external module can be shorter than the blank COMMON block in the calling program module, but all variables in the block in the calling program, from the first one up to and including the last variable significant to the external module, must be represented in both modules. You can freely omit named COMMON statements that are not needed in an external module.

Remember that in QB2 a COMMON statement must always appear before the first executable statement in a module. Each variable name in a COMMON statement must be of the same type as the corresponding variable name in the COMMON statement in the other module. Static arrays named in COMMON statements require that they be dimensioned in a DIM statement before the COMMON statement. Dynamic arrays named in COMMON statements require that the number of dimensions be indicated by a constant in the parentheses attached to the array name and that the DIM statement for them come after the COMMON statement.

Separately compiled modules may declare one or more than one subprogram. Since the SHARED attribute on the COMMON statements in the external module makes the listed variables global to all subprograms declared in the external module, you may find a need for the STATIC statement to keep certain variables private to some of the subprograms.

The QB2 manual suggests putting COMMON statements in an "include" file and using the $INCLUDE metacommand to include the include file in all the modules in which it is needed. This can be useful with COMMON statements for program chaining, but is of only limited usefulness for sharing variables with subprograms. The SHARED attribute is always needed in the external module and is sometimes not appropriate in the calling module. When you use named COMMON blocks, dif-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ferent named COMMON statements would probably be needed in different external modules.

Summary

Subprograms can be called from the main program or from within other subprograms. Variables in the main program and those in subprograms are hidden from each other unless the programmer makes them visible. Variables in the calling program can be passed to variables with different names in the subprograms, making it easy to call a subprogram with different sets of variables. Most methods of variable passing are two-way—the variables in the calling program are available to the subprogram, and changes to the corresponding variables in the subprogram change the value of the variables in the calling program. In one method only, that of passing with the argument lists, selected variables can be passed to subprograms without giving the subprogram the ability to change the value of the calling program variables (passing by value).

The methods of variable passing available in QB2 are as follows:

1. *Using the argument list of the CALL and SUB statements.* This is the most straightforward and the preferred method unless some consideration, such as a lengthy list of arguments, indicates another method. The same rules apply whether calling a subprogram in the same module or in a separately compiled program module in a user library. This method makes it easy to call a subprogram repeatedly with a different set of variables each time. Passing by value is permitted.

2. *Using the SHARED attribute in the main program.* Works only with subprograms in the same compiled module. Makes the listed variables shared with all subprograms in the module. Convenient when sharing with all of the subprograms is required. Invites possible conflicts with variables in some subprograms.

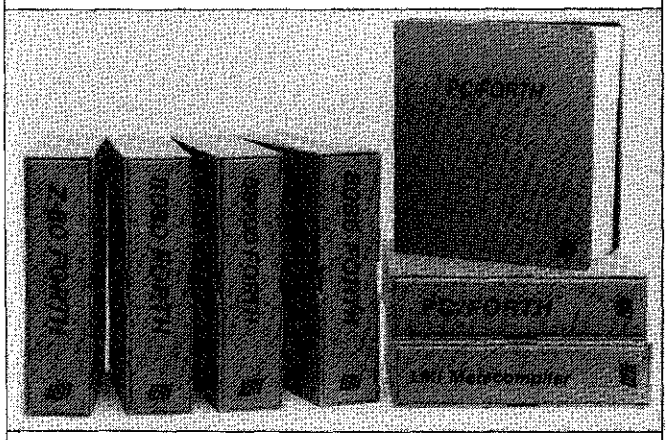
3. *Using the SHARED statement in subprogram declarations.* Works only between the main program and subprograms in the same compiled module having the SHARED statement. Does not cause sharing with other subprograms unless two or more subprograms share the same variable with the main program, in which case, they can share the variable with each other through the main program.

4. *Using paired COMMON statements.* Works only with subprograms in a separately compiled program module. Requires suitable COMMON statements in both the calling module and the external module. The SHARED attribute is required on the COMMON statements in the external module. Using paired COMMON statements and passing with the argument lists are the only methods of calling subprograms compiled separately and put in a user library.

These methods usually are combined in a program, with some variables being passed one way and some by others. □

Bernard H. Robinson, Jr., retired four years ago from a large aerospace company, where he was responsible for both hardware and software design for computer-controlled stations that tested company products. Since retiring, he has devoted his time to writing (he has been a frequent contributor to PJ) and consulting.

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler
- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:
- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New Version 5.0

PL/D Compiler

System language:

PL/D lets you access machine resources within a structured language. Move from high level to low level in a single source.

```
DAIR PL/D-88 ver 5.00 Jan 1, 1986 12:01
Line#  Org   Comm  Call
-----  ----  ----  ----
    1  0100  0000        ; PL/D example: zeroing memory
    2  0100  0000        SYS DEFINES ; read source library
    3  0100  0000          COMM  $H400 ; variables starts here
    4  0100  0400          BASE $HE000 ; .com file starts here
    5  0100  0400          ORG  $HE000 ; code gen  starts here
    6  E000  0400          ; define 8088 opcodes
    7  E000  0400          INSEQU  MOVESAX  $H8ECO ;
    8  E000  0400          INSEQU  MOVCXAX  $H8EBC8 ;
    9  E000  0400          INSEQU  MOVDIAX  $H8BF8 ;
   10  E000  0400          INSEQU  REPSTOSW $HF2AB ;
   11  E000  0400          INSEQU  CLD      $HFC ;
   12  E000  0400        ENDSYS ;
   13  E000  0400        IFNOT THEN ; skip subroutines
   14  E003  0400          EQU KS  64 ; 1kb in paragraphs
   15  E003  0400          EQU ZB  32 ; start 32k
   16  E003  0400          EQU ZE 640 ; end 640k
   17  E003  0400          EQU ZS  16 ; 16k at a time
   18  E003  0400          ;-------------------------------
   19  E003  0400          ; zero all memory in high level
   20  E003  0400          ; pl/d indents automatically
   21  E003  0400          ; note: ( ) is at compile time
   22  E003  0400          ;-------------------------------
   23  E003  0400          SUBR HILVLZRO ;
   24  E003  0408            DCL RMF RMC SEGP SEGC ; temps
   25  E003  0408            ( ZB * KS ) MOVESAX ; start
   26  E008  0408            FOR ( ZE - ZB / ZS ) -> SEGC DO ;
   27  E008  0408              0 -> RMF ;
   28  E014  0408              FOR ( ZS * KS / 2 ) -> RMC DO ;
   29  E01A  0408                0 # -> RMF ; indirect store
   30  E024  0408                RMF 2+ -> RMF ;
   31  E02C  0408              NEXT RMC ;
   32  E032  0408              SEGP + ( ZS * KS ) -> SEGP ;
   33  E03B  0408              MOVESAX ; new segment reg
   34  E03D  0408            NEXT SEGC ;
   35  E043  0408          ENDSUB HILVLZRO ;
   36  E044  0408          ;-------------------------------
   37  E044  0408          ; Speed optimize inner loop with
   38  E044  0408          ; same outer loop.
   39  E044  0408          ;-------------------------------
   40  E044  0408          SUBR FASTZRO ;
   41  E044  0408            DCL SEGP SEGC ; temps
   42  E044  040C            ( ZB * KS ) MOVESAX ; start
   43  E049  040C            FOR ( ZE - ZB / ZS ) -> SEGC DO ;
   44  E04F  040C              CLD ; forward direction
   45  E050  040C              ( ZS * 512 ) MOVCXAX ; word cnt
   46  E055  040C              0 MOVDIAX ;  ; indirect store
   47  E058  040C              0 REPSTOSW ; repeat store
   48  E05F  040C              SEGP + ( ZS * KS ) -> SEGP
   49  E066  040C              MOVESAX ; new segment reg
   50  E06A  040C            NEXT SEGC ;
   51  E070  040C          ENDSUB FASTZRO ;
   52  E071  040C        ENDIF ; execution starts below
   53  E071  040C  E003  HILVLZRO ; execute hi level
   54  E074  040C  E044  FASTZRO  ; do again
   55  E077  040C        EOF ;

NO ERRORS. Max table usage was  1% symbol and  0% macro

Line#  Name      Typ  Val    Reference line numbers
-----  --------  ---  -----  ----------------------
   11  CLD       Ins  00FC     44
   49  FASTZRO   Sub  E044     54
   23  HILVLZRO  Sub  E003     53
   14  KS        Equ  0040     25   28   32   42   48
    8  MOVCXAX   Ins  8EC8     45
    9  MOVDIAX   Ins  8BF8     46
    7  MOVESAX   Ins  8EC0     25   33   42   49
   10  REPSTOSW  Ins  F2AB     47
   24  RMC       Del  0402     28   31
   24  RMF       Del  0400     27   29   30   30
   24  SEGC      Del  0408     26   34
   41  SEGC      Del  040A     43   50
   24  SEGP      Del  0404     32   32
   41  SEGP      Del  0408     48   48
   15  ZB        Equ  0020     25   26   42   43
   16  ZE        Equ  0280     26   43
   17  ZS        Equ  0010     26   26   32   43   45   48
```

PL/D Example: clearing memory

Macros:

PL/D macros give you a powerful expression capability. Use global variables, object memory, compile time stack, or explicit parameter calls for communication.

Source:

PL/D comes in source and allows you to make the changes you need, when you need them. Find a hidden "gotcha" at the 80386 level? With PL/D, you are in control.

New features:

Symbol table now uses all available memory and listing includes a cross reference. PL/D for 8085 is included on Compiler disk.

Object **$49.95**
PLD compiler and printed user manual.

Compiler **$124.95**
PLD and P85 compilers, all source, and printed user and internals manuals.

NOT Copy Protected. For PC or compatible with MSDOS 2.0 or above, 192K, 128+ col. printer, and 5 1/4 inch floppy disk. Add tax in CA. Trademarks owned by manufacturers.

DAIR Computer Systems VISA MASTER CARD
3440 Kenneth Dr., Palo Alto, CA 94303

(415) 494-7081

Number 5 on the Reader Service Card

SugarCube Concurrent Computers
Hypercube Concurrent Environment
both from Intel Corporation

The SugarCube

Intel Scientific Computers has announced four desktop-sized models of its iPSC family of concurrent, hypercube architecture computers.

You can get a standard system for concurrent computing research, an extended memory system for artificial intelligence and symbolic computing applications, a vector system for numeric-intensive applications, and a hybrid version that combines numeric and symbolic capabilities within a single system. SugarCube systems start at less than $50,000.

Still here? There's more. Standard languages include Fortran, C, and Concurrent Common Lisp from Gold Hill.

Intel's hypercube machines range from 4 to 128 nodes in size. Each node is a complete single-board 80286/80287 computer containing its own memory and 8 point-to-point communication channels for passing messages between nodes. Applications written for 4 nodes are upwardly compatible.

The standard SugarCube system has eight nodes. The extended memory SugarCube-MX has 4 nodes, each with coprocessor, for a claimed combined peak performance of 26 million floating point operations per second (MFLOPS), 64-bit double precision. Performance for single precision 32-bit is 80 MFLOPS.

How close you can get to peak performance is a question—as any average massively-parallel processor walking the street will tell you.

Hypercube Concurrent Environment

Intel has announced a concurrent operating environment for its iPSC family of computers that is faster than previous versions and adds concurrent debugging, multiple user access, connection to Unix-based front ends, and support for "hybrid" iPSC systems (which combine numeric and symbolic processing into one system).

Now hypercubes come with the same set of tools that conventional systems have had for years, says Charlie Bishop, marketing manager for Intel Scientific Computers.

The new operating system is resident on each processing node of the machine and has a 10-fold increase in message-passing performance.

Continued...

Tom Rettig's Library™

Prewritten Solutions to Programming Problems

Clipper Edition

Advanced Extended Library

- Convenient .LIB library file to link
- Separate object files so only the code actually used is added to your program
- Optimized for speed and code size
- Requires Clipper, Winter '85 or later

dBASE III PLUS Edition

Advanced Programmer's Library

- Interface to C/assembler is fully compatible with Clipper's Extend system
- Functions are CALL procedures
- Requires dBASE III PLUS, any version, and 128K of additional memory

Each edition comes complete with...

- Entire source code, nothing withheld. Cleanly written, liberally commented, easy to modify, helpful for learning.
- Support by phone and electronic mail
- No royalties or copy protection
- Documentation inserts for dBASE or Clipper manual. Accurate, complete, easy to use, page-per-command.
- Handy plastic reference card
- Full money-back guarantee

Over 175 functions; 55% written in C, 35% in Assembler, 10% in dBASE

$99.95 per edition at dBASE/Clipper dealers or direct

II/III Tom Rettig Associates
Excellence in dBASE since 1982

9300 Wilshire Boulevard, Suite 470
Beverly Hills, CA 90212-3237, U.S.A.
(213) 272-3784 ■ Telex: 4996426 RETTIG
Source: BCR480 ■ CompuServe: 75066,352

dBASE and dBASE III PLUS are Ashton-Tate trademarks, Clipper is a Nantucket trademark

Call or write for free product information

Number 117 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Software Family

649 Mission Street
San Francisco, CA 94105

(1)800-443-7176

In California or outside U.S.

(415)583-4166

Trust TSF to provide the best value and service!

* Technical advice and support by programmers
* Honest and equitable business practices
* Prompt, flexible service to meet your needs

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on software orders over $100 ($3 delivery charge on orders under $100). We add actual charges for UPS Blue Label or Federal Express rush service. Our COD fee is $2. We allow return privileges on most products. We carry a large inventory and ship promptly, so you receive your shipment within 3 to 6 working days of placing your order. Give us a try.

__ Please send me a free catalog.
I'm most interested in:
__ Basic __ C __ Pascal
__ Turbo Pascal __ Other Languages
__ dBase __ Technical Publishing
__ Please send the following products:

Name: _____

Address: _____

City/State/Zip: _____

Credit Card: _____

Expiration Date: _____

Mail to: TSF, 649 Mission Street, San Francisco, CA 94105

Basic C
Turbo Pascal dBase
Publishing AI

Aldebaran **Source Print** (list $75)	**$59**
Blaise **C Tools +** (list$175)	**$125**
Blaise **Light Tools** For Datalight (list $100)	**$83**
Blaise **Turbo Power Tools+** (list $99)	**$79**
Borland **Turbo Basic** (list $99)	**$68**
Borland **Turbo C** (list $99)	**$68**
Bytel **Genifer** dbase III application generator ($395)	**$250**
Comtel **dbScope** Interactive dBase debugger ($70)	**$62**
Comtel **dbTools** (list $70)	**$62**
Creative **Vitamin C** Specify compiler (list $225)	**$159**
Creative Programming **VC Screen** (list $100)	**$79**
Datalight **Developer's Kit** (list $99)	**$75**
Datalight **Optimum C** w/Easy Edit (list $140)	**$115**
Fox **FoxBase+** fast dBase compiler ($395)	**$330**
Gimpel **C-Terp** Specify compiler (list $300)	**$224**
Gimpel **PC-Lint** (list $139)	**$103**
Greenleaf **Data Windows** Specify compiler ($225)	**$157**
Guidelines **C++** for Microsoft C (list $195)	**$175**
Lattice **C** (list $500)	**$270**
MicroHelp **Mach2 for Microsoft Basic** ($69) windows, fast i/o, critical error trapping, more!	**$55**
MicroHelp **Mach2 for Turbo Basic** ($69)	**$55**
MicroHelp **Mach2 for Turbo Pascal** ($69)	**$55**
Microsoft **C w/Codeview** (list $450)	**$270**
Microsoft **Fortran w/Codeview** (list $450)	**$270**
Microsoft **Quick Basic** (list $99)	**$65**
MKS **Toolkit** Korn shell, vi, grep, 30 more ($139)	**$115**
Periscope **Periscope I** (list $345)	**$279**
Periscope **Periscope II** (list $175)	**$139**
Periscope **Periscope III** (list $995)	**$799**
Phoenix **PFix-86+** (list $395)	**$250**
Phoenix **Plink-86+** (list $495)	**$315**
Tom Rettig **Library** (list $99)	**$87**
Turbo Power **Extender** (list $85)	**$68**
Turbo Power **Optimizer** (list $75)	**$59**
Turbo Power **Optimizer** w/Source (list $175)	**$99**
Turbo Power **TDebug Plus** (list $60)	**$48**
Turbo Power **Turbo Utilities** w/source ($95)	**$76**
Wallsoft **UI Programmer** (list $295)	**$260**
Wordtech **dBXL** dBase III work-alike (list $169)	**$110**
Wordtech **Quicksilver** dBase compiler (list $599)	**$459**
Xerox **Ventura** best available DTP (list $895)	**$599**

TSF carries hundreds of products from dozens of publishers. Call or write for a FREE catalog or a price quote.

Number **98** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Instant Replay™
Version II

Demonstrations
Presentations
Prototypes
Tutorials
Menus
Music

"Indispensable . . . A clear improvement over Dan Bricklin's Demo Program."
PC Magazine

"Incredible . . . We built our entire Comdex Presentation with Instant Replay."
Panasonic

"Instant Replay brings new flexibility to prototypes, tutorials, & their eventual implementation."
Electronic Design

"I highly recommend Instant Replay."
Computer Language

Build tutorial replays of actual programs, or animated prototypes. Memorize and replay keystrokes and pause times. Has the unusual ability to insert prompts, pop-ups, prototypes, user involvement, music, and branching menus into replays.
Includes Screen Maker, Keystroke/Time Editor, Prototyper, Text Editor, Music Maker, Menu Maker, Screen Grabber, Animator, Control and Insertion Guides, 250 Page Manual, Online Tutorial, (Not Copy Protected, No Royalties), 60 Day Satisfaction Money Back Guarantee.

***Instant Replay II* $149.95**
Demo Diskette $5 Dealer Poster $3

Call or Write. We accept Visa, AMEX, Master Card, COD, PO. Dealer Inquiries Welcome.
Outside US pays postage.

Nostradamus
3191 South Valley Street (ste 252)
Salt Lake City, Utah 84109
(801) 487-9662
Number 112 on the Reader Service Card

With the concurrent debugger, the programmer can observe and control message behavior in the system and can analyze system-wide communications flow for tuning performance. Also supported are the familiar features of a sequential debugger including high-level language breakpoints. Simultaneous debugging activity is possible in every node of the system.

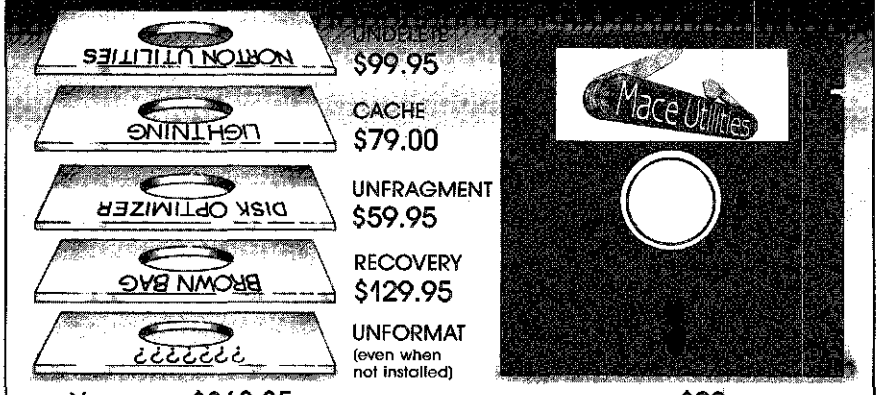
A programmer can use a Sun-3 or similar workstation to simultaneously edit files, debug, cross-compile, and control program execution, all in separate windows.

The Hybrid Hypercube
In hybrid operation, the hypercube can be populated with a mixture of node types. A programmer uses a common set of message-passing functions to implement applications using a mixture of languages, sending messages between programs written in C, Concurrent Common Lisp, and Fortran.

To illustrate hybrid operation, Intel refers to an application that would require the computation of massive amounts of sensor data, coupled with a need for automated analysis or expert system control.

CGI for OS/2
GSS
P.O. Box 4900
Beaverton, OR 97075
(503) 641-2200
$495 now, $695 later
Prior to IBM's April 2nd announcement, rumors were that IBM would chose Metagraphics, the fast PC graphics metalanguage, for its new machines. They picked GSS instead. GSS has announced that for the fifth time, IBM has chosen CGI as its device-independent graphics platform for microcomputers. To provide graphics support for IBM Operating System/2 and the new hardware, IBM has announced a new version of the "IBM Graphics Development Toolkit." GSS followed the announcement with GSS Toolkit Version 3.0.

ONE TOOL DOES IT ALL!



You pay $368.85 or $99

To Order Call 800-523-0258

★★★★★ *SUPERIOR*
"Mace+Utilities is worth its weight in gold."
80 Micro, Jan. 1987

"I advise you to run (don't walk) to the phone or post office and order a copy of the Mace+Utilities."
John Dvorak, InfoWorld, Mar. 24, 1986

"The Mace Utilities is a package of indispensable tools for serious hard disk users. We recommend every PC owner with a hard disk keep a copy ready at hand."
InfoWorld, Feb. 16, 1987

"What are you waiting for?"
PC Magazine, Mar. 31, 1987

Paul Mace
SOFTWARE
123 N. First St., Ashland, OR 97520

Disk Optimizer, Norton Utilities, Lighting, and Brown Bag are trademarks of Soft Logic, Peter Norton Computing, PCSG, and The Software Resource Group, Inc.

Number **80** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

3.0 is a superset of the graphics available in DOS-based 2.0, and it includes support for IBM's new C/2, Pascal/2, FORTRAN/2, BASIC/2 and Macro Assembler/2.

However, because of the major architectural changes of OS/2, the software interrupts used by the DOS version have had to be replaced with a dynamically linked library architecture, and new device drivers are also required.

The Parallel Solution

We've been looking at a very interesting machine that for only $1500 (for the four-floppy-drive machine) appears to offer the answer to people's crying need for multi-tasking. This machine will run two DOS applications at once, and, contrary to all other multi-tasking solutions now available, runs each at the same full speed that can be obtained on a single tasking machine.

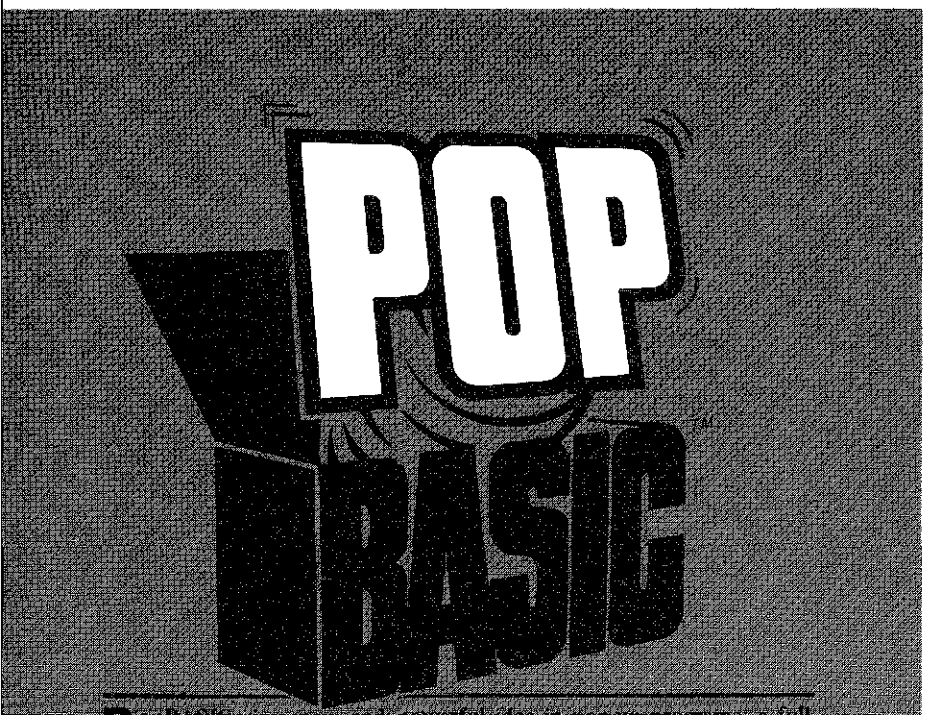
Furthermore, each task runs in the foreground, and both can be worked on and modified simultaneously.

Using the unit has just about doubled our work throughput.

The monitor that displays the two applications (included in the $1500) is fabulous: 160 characters by 25 lines, or alternately, 80 characters by 50 lines. The total display space appears to be nearly twice that of a normal monitor. Characters are full-formed and legible. This is much better than putting an application in a window on a standard monitor.

The unit will communicate between applications too. Information exchange between concurrent applications could be faster, but we actually have found little need so far to exchange information.

The biggest drawback is the machine's size: just about twice the size of a normal DOS machine. We just cleared off some of our desk, though, and it fits easily. □



PopBASIC is a new and powerful idea in pop-up programs: a full-featured interpretive BASIC that's memory resident and available at all times. It's more than a utility, it's a utility creator. You can create the perfect pop-up programs, the way you want them: smart keyboard enhancers, notepads, demo aids, calculators, alarm clocks, and hundreds of others that can't be done with any other product. Anything you can program in BASIC can become a pop-up utility.

And you don't have to start from scratch. PopBASIC comes with programs for a complete pop-up system:

- Notepad
- Calculator
- Alarm Clock
- Keyboard Enhancer
- Large File Peeker
- ASCII & Key Code Tables

If you're using any other pop-up software, PopBASIC's flexibility will knock your socks off. If you're new to pop-ups, start with the one you won't outgrow. HURRY, ORDER NOW!

ONLY $79.95*
CALL (818) 243-2235

HEDGE SYSTEMS

511 West Glenoaks Blvd., Suite 230, Glendale, CA 91202

*Price includes shipping to all US cities. CA orders add sales tax.
System requirements: IBM PC/XT/AT or true compatible, PC-DOS/MS-DOS 2.0 or later, 192K RAM.
MS-DOS is a registered trademark of Microsoft Corp. XT and AT are registered trademarks of International Business Machines Corp.

Number 39 on the Reader Service Card.

2 Programmers & Developers New Products!

Distribute Your Demos with No Royalties

Screen Machine creates interactive demos, tutorials, menu systems and DOS shells. Includes a text screen editor that optionally generates source code* and binary or text files. Never write code for screen display again. Capture any program's text screens for editing and your own use. Capture CGA compatible graphics screens for BLOAD or direct display. SAVE hundreds of HOURS of work.

Now there's no need for separate screen and demo software packages and no need to pay outrageous royalties. Priced at only $79.00.

*Turbo Pascal, Mach 2 for Turbo, Assembler, dBASE II & III, BASIC (including The Inside Track and Mach 2).

Supercharge Turbo Pascal

Mach 2 for Turbo Pascal adds assembler speed to your programs. 90+ subroutines, most in assembler, give you speed and functionality you never knew was possible. No knowledge of assembler language required.

INSTANT displays. INSTANT windows (incl. exploding and boxed). FASTEST sort you've seen. Read/write files FAST as DOS. INSTANT menus, 1-2-3 horizontal and vertical bar.

Trap ^C/^Break & DOS critical errors so no more A)bort, R)etry or I)gnore. Emulate BASIC PRINT USING for FAST formatted numbers. Execute any prog, batch or DOS command without ending program.

Read environment. Read file directory. Get/set file attributes. Plus too many string functions to describe here. No royalties when you distribute COM programs. All source code included. A true bargain at $69.00.

NOT COPY PROTECTED. 30 Day Money-Back Performance Guarantee. Requires IBM/compatible & DOS 2+.

Order Now 800-922-3383

We welcome VISA/MC. COD US only $3. S/H US $3, Canada $5, Elsewhere $18. GA res. add tax and call 404-973-9272. Demo available. Send $5 check. Refunded on direct purchase.

We also publish Stay-Res, Mach 2 for BASIC, The Inside Track and Peeks 'n Pokes.

MicroHelp, Inc.
2220 Carlyle Drive
Marietta GA 30062

Number 90 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ BILLBOARD

***PJ Billboard** is a directory of products and services of interest to the IBM PC programmer. The goal is to provide our readers with a wide range of information on programming products, introductions, and updates.*
***PJ Billboard** serves advertisers by providing a low cost, flexible format in which they can present their products to PJ readers.*
*The standard layout is four columns, 9 picas wide, with copy set in 7 point on 9 point leading. A **PJ Billboard** listing offers the advertiser a 400-character product description including company name, address, and telephone number. The listings are $89, prepaid, non-commissionable, for a one-time insertion. Listings must be received typed or legibly printed, with boldface or italic words indicated where desired. A convenient order form is available upon request. Copy deadlines are the same as the normal advertising space reservation dates, which can be found on our rate card.*

For more information call the
Programmer's Journal
Advertising Department

(503) 484-2162

FREE PUBLIC DOMAIN SOFTWARE FLYER!

IBM & Compatibles (MS-DOS) More than 4000 programs on 165 disks for only $3.50 per disk! Games, Educational, Business, Personal, Utility and Miscellaneous Software! Terrific software and service at unbelievable prices!

Send for FREE FLYER to:

AP-JP Enterprises, Inc.
P.O. Box 1155
W. Babylon, NY 11704

UNIX / XENIX SYSTEM 5

Compatible CURSES window and Keyboard library for MS-DOS and MS OS/2 C compilers. Create, paint, scroll, pop-up, pull down, title and overlay windows with no hassle. Complete keyboard, function key and key-pad translation. Fully optimized screen output with device independence — display an entire screen in a fraction of a second.
BEST PRODUCT — COMPUTER LANGUAGE — JUNE '87

Price: Object Library - $119.00
with Source Code - $289.00

Aspen Scientific
P.O. Box 72
Wheat Ridge, CO 80034-0072
303-423-8088

POCKET REFFERENCES FOR C PROGRAMMERS, UNIX AND DOS USERS

C Library Reference — $6.00
MS-DOS Reference — $3.00
UNIX Command Summary —$6.00
Exploring UNIX Environment — $16.95
(Specify System V, BSD 4.2/4.3, or XENIX V)

Other references, UNIX software, and consulting available.

To order or send for our free catalog write or call:

SSC, Inc.
P. O. Box 55549-P
Seattle, WA 98155
(206) 367-UNIX

HOT PROGRAMMING UTILITIES!

This issue of ***Programmer's Journal*** is packed full of advertisers offering useful utilities to help with your programming. When you call a company about their product, be sure to let them know you read about it in ***PJ.***

Programmer's Journal
P. O. Box 30160
Eugene, OR 97403
503-484-2162

CONVERT BASIC TO C or PASCAL

BAS_C™ converts BASICA to C which can be directly compiled by any C compiler.
BAS_PAS™ converts BASICA to Turbo Pascal™. Both BAS_C™ and Bas_PAS™ transform the spaghetti code to the scoped, indented, structured program automatically. Deadcode. Graphic, Random file.

Sample Conversion: send your own BASIC program (1¢/1 line) and receive the printout.

Demo disk $5.00

Gotoless Conversion
P.O. Box 50068
Denton, TX 76206
214-221-0383

FREE C-DISPLAY LIBRARIAN™

During our Summer Special, we'll include a FREE C-Display Librarian™ with every DisplayExpress™ purchased at the regular price. See our ad on the next page for more information.

SYDETECH
43-23 Colden St. Suite 17C
Flushing, NY 11355
718-886-0221

ADVERTISER INDEX

Reader Service Number	Advertiser Name	Page Number
116	Aker Corporation	33
86	Aspen Systems	21
30	Austin Code Works	41
51	Az-Tech Software, Inc.	34
36	B C Associates	5
2	Blaise Computing	3
3	Borland International	C4
38	C Ware	8
56	CompuView Products, Inc.	11
*	Creative Programming Consultants	40
83	Crescent Software	55
5	Dair Computer Systems	60
103	Desktop Ai	48
70	E.C.I.	57
8	Ecosoft, Inc.	1
113	Elfsoft	37
118	Fleming Software	41
*	Gimpel Software	43
57	Hawkeye Grafix	54
39	Hedge Systems	63
58	Interface Group, The	9
91	King Computer Services	16
63	Laboratory Microsystems, Inc.	59
44	Lahey Computer Systems	52
80	Mace Utilities	62
92	Manx Software Systems	27
69	MDS, Inc.	6
45	MetaWare	17
90	MicroHelp	63
13	Micromotion	10
104	MicroWay	25
94	MMC AD Systems	41
109	NanoSoft	45
112	Nostradamus	62
16	Opt-Tech Data Processing	34
6	Periscope Company, The	15
114	Programmer's Shop	49
*	Programmer's Journal	56
82	PS Business Systems	21
*	Quaid Software Limited	21
97	Qualstar Corporation	54
46	Quilt Computing	57
20	Rational Systems	28
93	Software Connection	48
98	Software Family, The	61
108	Software Garden, Inc.	37
115	Software Interphase, Inc.	8
110	Softway, Inc.	53
47	Solution Systems	29
85	Sunny Hill Software	7
74	Sydetech, Inc.	C3
117	Tom Rettig Associates	60
35	Turbo Source Search	41
77	Wendin, Inc.	53
73	Whitewater Group	C2

** Contact these Advertisers directly*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Summer Special

FREE C-Display Librarian™ with every DisplayExpress™

Here's the quick, easy way to make screens, create demos and write better programs

"If you want to include bit-mapped graphics in your application, then this might be the package for you"
(D. Drumheller and M. Wilson, Computer Language)

"DisplayExpress and C-Display Librarian are very fine products and I am looking forward to incorporating them into my applications development"
(R.W. Him, Vice-President, NHDS Inc., Chicago, IL)

"DisplayExpress and C-Display Librarian are a great time saver and should be in the tool-kit of every C language programmer"
(D. Wilson, McLean, VA)

DisplayExpress™ $99

DisplayExpress™ helps you make screens and demos quickly and without programming. Call such screens from your programs or use them for your demodiskettes, marketing shows and continuously running demos. VGA, EGA, CGA, colors, character graphics, interactive screen design, and much more! Free source for the run-time support! No royalties!

C-Display Librarian™ ~~$99~~ FREE with every purchase of DisplayExpress™

The easy way for you to write great programs! The C-Display Librarian™ helps you call BIOS video functions, control the screen, perform data entry and validation, create graphics and much more. Specifically designed for display management and interactive programming. Support for VGA, EGA and CGA. Full source included. No royalties!

Send check or money order to:

SYDETECH
SYSTEM DEVELOPMENT TECHNOLOGIES INC.
43-23 COLDEN STREET, #17C, FLUSHING, NEW YORK 11355
PHONE: (718) 886-0221

Available for IBM Personal System/2 and PC/XT/AT or TI Pro/Business-Pro computers. Compilers supported: Microsoft MS-C 5.0, MS-FORTRAN 4.0, MS-Pascal 3.3 and Lattice 3.1 or higher.
©1987 **SYDETECH** System Development Technologies Inc.

Number **74** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NEW

Turbo C®

NOW SHIPPING!

Turbo C: The fastest, most efficient and easy-to-use C compiler at any price

Compilation speed is more than 7000 lines a minute, which makes anything less than Turbo C an exercise in slow motion. Expect what only Borland delivers: Quality, Speed, Power and Price.

Turbo C: The C compiler for amateurs and professionals

If you're just beginning and you've "kinda wanted to learn C," now's your chance to do it the easy way. Like Turbo Pascal, Turbo C's got everything to get you going.

If you're already programming in C, switching to Turbo C will considerably increase your productivity and help make your programs both smaller and faster. Actually, writing in Turbo C is a highly productive and effective method—and we speak from experience. Eureka: The Solver™ and our new generation of software have been developed using Turbo C.

Turbo C: a complete interactive development environment

Free MicroCalc spreadsheet with source code

Like Turbo Pascal® and Turbo Prolog,® Turbo C comes with an interactive editor that will show you syntax errors right in your source code. Developing, debugging, and running a Turbo C program is a snap.

Turbo C: The C compiler everybody's been waiting for. Everybody but the competition

Borland's "Quality, Speed, Power and Price" commitment isn't idle corporate chatter. The $99.95 price tag on Turbo C isn't a "typo," it's real. So if you'd like to learn C in a hurry, pick up the phone. If you're already using C, switch to Turbo C and see the difference for yourself.

System Requirements

IBM PC, XT, AT, PS/2 or true compatibles. PC-DOS (MS-DOS) 2.0 or later. One floppy drive. 384K RAM.

Technical Specifications

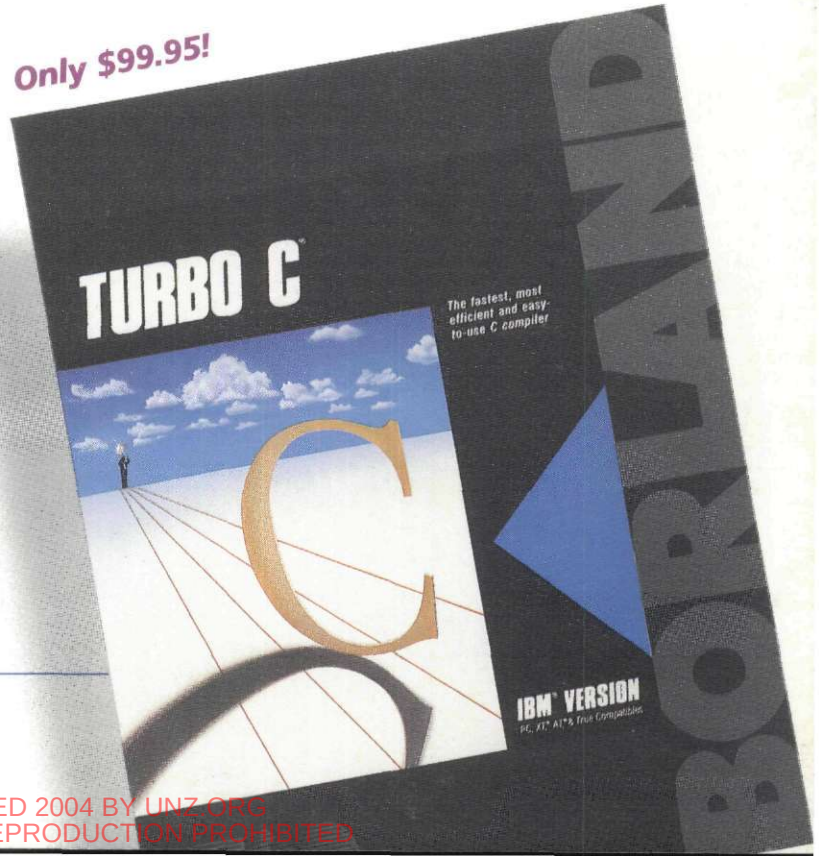
- ☑ Compiler: One-pass compiler generating linkable object modules and inline assembler. Included is Borland's high performance "Turbo Linker." The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).
- ☑ Interactive Editor: The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.
- ☑ Development Environment: A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pull-down menus and windows.
- ☑ Links with relocatable object modules created using Borland's Turbo Prolog into a single program.
- ☑ ANSI C compatible.
- ☑ Start-up routine source code included.
- ☑ Both command line and integrated environment versions included.

Turbo C, Turbo Prolog and Turbo Pascal are registered trademarks and MicroCalc, Turbo Linker and Eureka: The Solver are trademarks of Borland International, Inc. Microsoft C and MS-DOS are registered trademarks of Microsoft Corp. IBM, XT, and AT are registered trademarks of International Business Machines Corp. Copyright 1987 Borland International BI-1104C

Sieve benchmark (25 iterations)

	Turbo C	Microsoft® C
Compile time	3.15	22.41
Compile and link time	6.55	29.49
Execution time	6.59	10.11
Object code size	239	249
Execution size	5748	7136
Price	$99.95	$450.00

Benchmark run on a 6 Mhz IBM AT using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51.



BORLAND INTERNATIONAL
Vive la différence

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Number 3 on the Reader Service Card

For the dealer nearest you or to order by phone call

(800)255-8008

in CA (800) 742-1133 in Canada (800) 237-1136

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED